510.84
Il6r
no. 895
p 2

Math

UIUCDCS-R-77-895 UILU-ENG 77 1749

BURSTLOGIC: DESIGN AND ANALYSIS OF LOGIC CIRCUITRY TO PERFORM ARITHMETIC ON DATA IN THE BURST FORMAT

By

LEON CLEMENS TIETZ

May, 1977

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-77-895

BURSTLOGIC: DESIGN AND ANALYSIS OF LOGIC CIRCUITRY TO PERFORM ARITHMETIC ON DATA ON THE BURST FORMAT

By

Leon Clemens Tietz

May, 1977

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1977

BURSTLOGIC: DESIGN AND ANALYSIS OF LOGIC CIRCUITRY TO PERFORM ARITHMETIC ON DATA IN THE BURST FORMAT

Leon Clemens Tietz, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1977

The goal of this research was to develop designs for arithmetic processors with input and output data both in the Burst format. An additional requirement was that these designs be constructible from readily available logic circuits such as TTL SSI and MSI and be competitive on both a cost and performance basis with the Block Sum Register (BSR) techniques first proposed for Burst Processing.

The result is a family of arithmetic processors based on a design for adding Bursts known as the Perverted Adder (PA). The PA is a full adder and a flip-flop connected with the sum output of the full adder fed back to the carry input via the flip-flop. This simple circuit performs Burst addition. However, the output Bursts are uncompacted, i.e. the ones are no longer adjacent to one another, whereas BSR designs output compacted Bursts.

Designs for a multiplier and divider use a tree of perverted adders to perform the requisite additions in parallel. Several modifications of this basic PA tree are also presented. They permit a reduction in the amount of hardware and a natural extension to Burst lengths other than the integer powers of two that are so natural for PA tree design. Several designs are also presented for performing arithmetic on signed Bursts. Designs employing both the sign and magnitude and a biased Burst technique are illustrated.

The designs for unsigned Bursts and some of those for signed Bursts use no registers to assemble Bursts before or after the processor. This design feature makes these truly on-line processors and gives them better high frequency response for a given clock rate than the other logic designs which have been proposed.

A PA CPU has been constructed to demonstrate Burst multiplication and division using PA techniques. The design of the CPU is basically a PA divider with modifications to permit the circuit to function as a multiplier. The design is based on an 8-bit-per-Burst format and displays the output in the form of the number of "1" bits in either 8 or 64 time slots at the user's option.

A comparison to other Burst processor designs shows that the PA designs are less hardware intensive and are generally faster than the other designs.

ACKNOWLEDGMENT

I am deeply indebted for the generous assistance given me during the preparation of this thesis. I wish to thank Frank Serio and Stan Zundo for their assistance in the preparation of the hardware needed for this thesis. Many thanks to Marty Blakey for preparing the drawings and to Nadine Luc for typing the manuscript. Their long hours are greatly appreciated.

To my advisor, Professor Poppelbaum, go special thanks for his ideas, guidance and encouragement.

Last but not least, thanks to my mother and father for their encouragement and to my wife, Addie, for her patience and understanding.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 An Explanation of Burst Processing

Burst processing was first proposed in 1974 by Professor W. J. Poppelbaum (1) as a PCM transmission and processing system that combined the advantages of both stochastics and weighted binary methods. The unit of information in Burst processing is the Burst which consists of a predetermined number of "bits". These bits are all of equal weight as opposed to weighted binary where bit position determines a weighting factor.

The equal weighting of the bits in a Burst causes Burst processing to have high noise tolerance without a need for sophisticated error correcting codes and the complex hardware associated with such devices. This noise tolerance does not come cheaply. A ten bit Burst is able to represent 11 different values. Only about 3-1/2 bits of weighted binary are necessary for the same precision. Another advantage is that there is no need for synchronization. Because every bit has the same weight you can start anywhere.

Because Bursts are a deterministic representation of values, they do not suffer from the severe problems of attaining accuracy that stochastic methods do. In Burst processing the accuracy is directly proportional to the length of the bit stream, or number of Bursts being observed. In stochastics the accuracy goes up only as the square root of the length of the bit stream.

The first method for performing arithmetic on Bursts involved a Block Sum Register (BSR). A BSR is a shift register with each bit position of the register controlling a current source (see Figure 1). As the data in Burst format passes through the BSR it produces an output signal proportional to the number of one bits in the register. BSRs perform arithmetic in a quasi-analog fashion and re-encode the resultant data to Burst format for retransmission.

Applications of Burst processing have been demonstrated by the Information Engineering Laboratory, under the direction of Professor Poppelbaum. (2) Many current projects are using Perverted Adder (PA) techniques as developed in this thesis rather than BSRs or other logic approaches to performing arithmetic. Comparisons of the various methods are found in chapter 4.

1.2 Burst Encoding Methods

Before data can be transmitted or processed in the Burst format it is necessary to encode it in that form. The simplest form of encoder is the ramp encoder. A staircase waveform is generated and constantly compared to the analog input. If the analog signal is greater than the staircase, the encoder transmits a one. If it is lower, a zero is transmitted. Figure 2 depicts this encoding process. This simple encoder is incapable of generating a Burst sequence with a long term accuracy greater than that available from a single Burst (e.g. one part in ten).

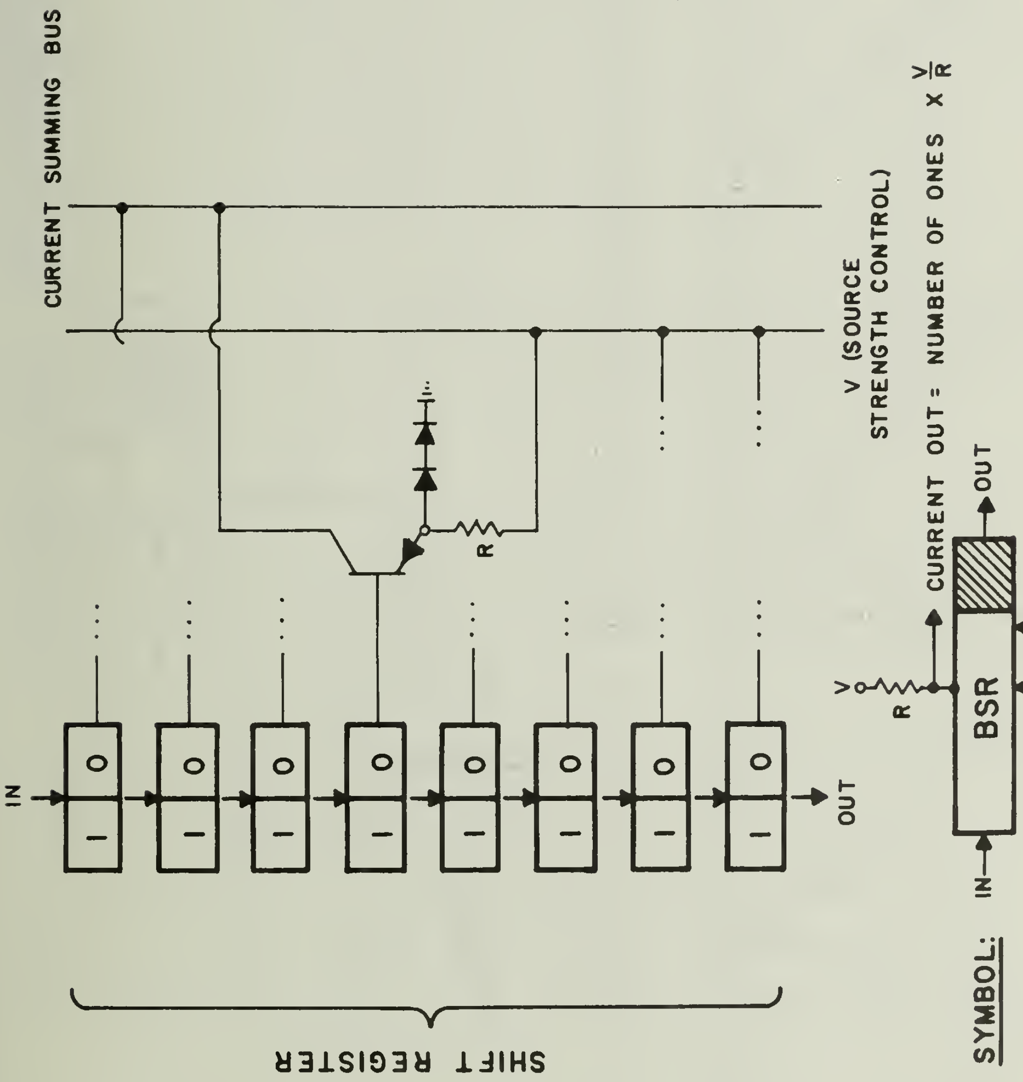


Figure 1. Construction of an 8-bit BSR.

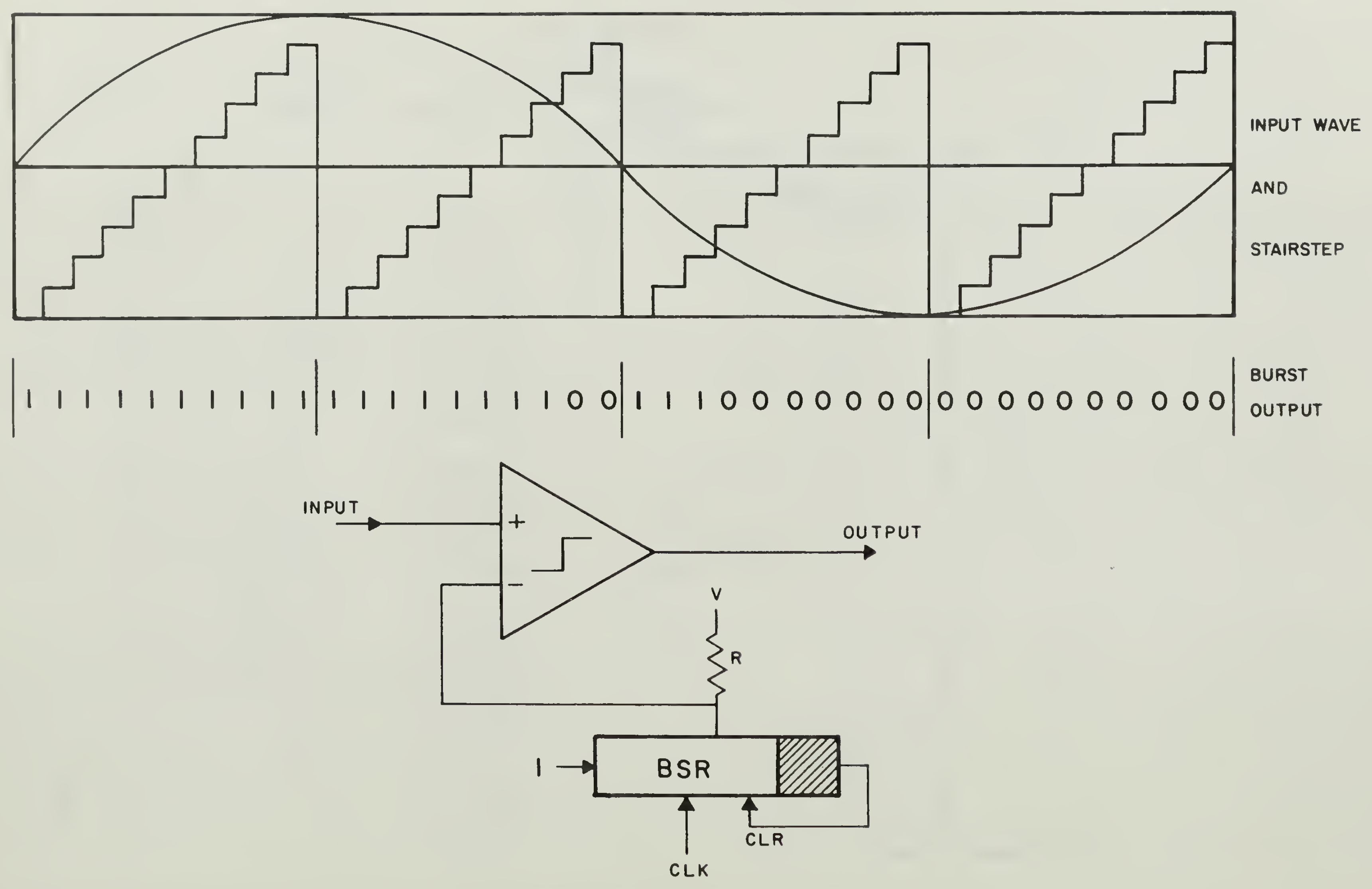


Figure 2. Ramp encoder principle of operation and circuit.

The vernier encoder yields results which are more accurate when averaged over a longer period of time. The principle is to take an n step ramp encoder and augment it with a second staircase generator operating at 1/n the frequency and with a stepsize of 1/n that of the first encoder. This scheme is capable of one percent precision in ten Bursts of ten bits each. Figure 3 shows the construction of such an encoder and the output while encoding a number with 1% precision. The output shown is the number of ones in each Burst generated by the encoder.

An encoder developed by Taylor (3) has superior high frequency response and improved noise characteristics. The complexity of this Delta Block Encoder (DBE) is comparable to the ramp encoder. A significant difference is that the output is in the form of uncompacted Bursts (see section 1.3). Uncompacted Bursts are perfectly acceptable to BSR or PA designs, but are incompatible with the other arithmetic processors discussed in chapter 4.

The DBE is similar to conventional delta modulation. The input signal is compared to the encoded version and the next output is chosen to minimize the error. The trade-off between frequency response and precision is done by choosing the length of encoded signal used in the comparison process.

1.3 Definitions

A <u>compacted Burst</u> is a Burst as the principle of Burst processing

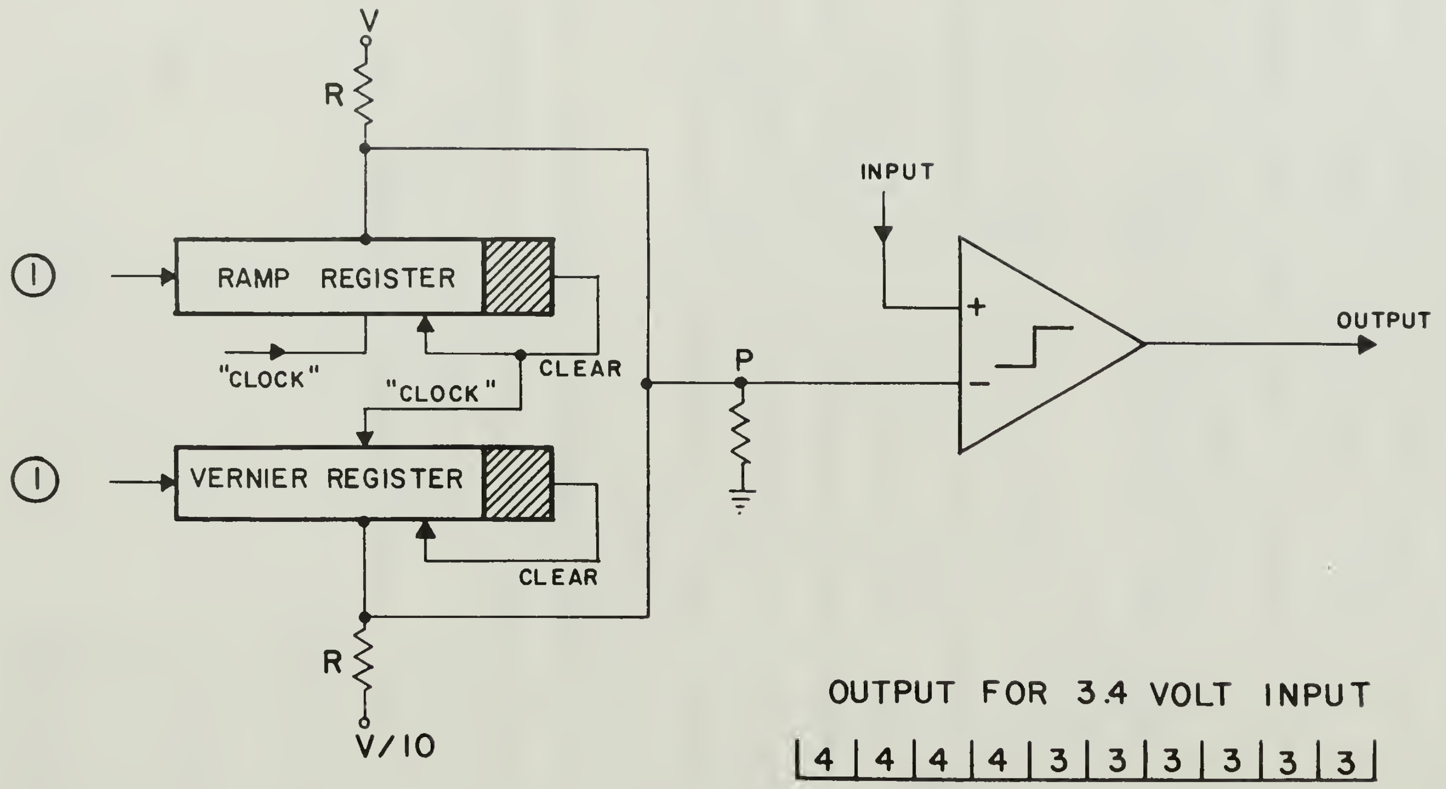


Figure 3. Vernier encoder and an example of Burst values for an encoded input.

was developed. All the ones in a single Burst are adjacent. In this representation the beginning of a Burst can easily be defined as the first one of the adjacent group.

Uncompacted Bursts have ones scattered throughout the Burst and thus lose the sense of a "burst" of ones. Better frequency response is attainable with uncompacted Bursts, but some of the cheap error correcting methods (3) possible with compacted Bursts are lost.

A Super Burst is a group of n Bursts each n bits long. If a Burst then can have a precision of p, a Super Burst can have a precision of p^2. The vernier encoder of section 1.2 is designed to produce Super Bursts.

The Bursts that have been described thus far are capable of representing only positive numbers. They will be referred to as unsigned Bursts. Most Burst applications have used unsigned Bursts.

Representation of negative numbers has been accomplished in two ways. The sign and magnitude representation uses an additional signal to represent the sign of the number, the magnitude is the same form as an unsigned Burst. A biased Burst represents signed numbers by assigning the value of zero to a half filled Burst; less than half is negative, more than half is positive. Designs employing both representations are presented in chapter 2.

2 PERVERTED ADDER CIRCUITS

2.1 Introduction

A new family of circuits has been designed to perform arithmetic on data in the Burst format. This family is the result of a search for a logic design that is competitive with the Block Sum Register (BSR) arithmetic elements that were first proposed for Burst arithmetic.

This family of circuits is based on a simple circuit building block which has been called the Perverted Adder (PA). The name was chosen because of the unconventional use of a full adder in the circuit. It should be noted that logically simpler realizations of a PA than the one shown here are possible. However, their realization with conventional SSI and MSI chips yields a higher chip count than the circuit described here.

2.2 Logic Description of the PA

The function of the PA is to accept input in Burst format on two input lines and to produce as an output the sum, also in Burst format. This sum must necessarily be scaled by a factor of two to prevent overflow. Table 1 is a state table (4) which describes the PA. Notice that there are two inputs "X_1" , and "X_2", two internal states, and one output.

Table 1: State table for the Perverted Adder (PA)

(PS = Present State; NS = Next State)

	NS,Z			
PS	X_1X_2 = 00	01	11	10
A	A,0	B,0	A,1	B,0
B	B,0	A,1	B,1	A,1

The machine is in state "A" when an even number of pulses have arrived on the two inputs and in state "B" after an odd number of pulses. An output "1" is generated whenever the machine goes from state "B" to state "A" or when two ones appear at the inputs simultaneously.

The combinational equations describing this machine are:

$$Z = x_1x_2 + x_1y + x_2y \qquad (1)$$

$$Y = \overline{x}_1\overline{x}_2y + x_1\overline{x}_2\overline{y} + \overline{x}_1x_2\overline{y} + x_1x_2y \qquad (2)$$

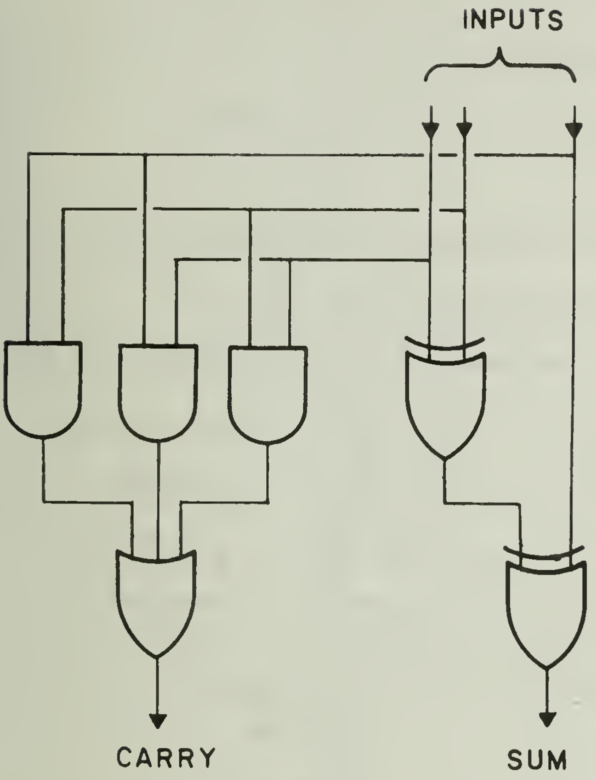
"Z" is the output and "Y" is fed back to the delay element yielding "y" when the flip-flop passes the signal. Except for an interchange of the left hand sides, the above equations are just those of a conventional serial adder. Figure 4 shows the logic of a full adder, the full adder and a D flip-flop connected as a Perverted Adder and the symbol used to represent the PA.

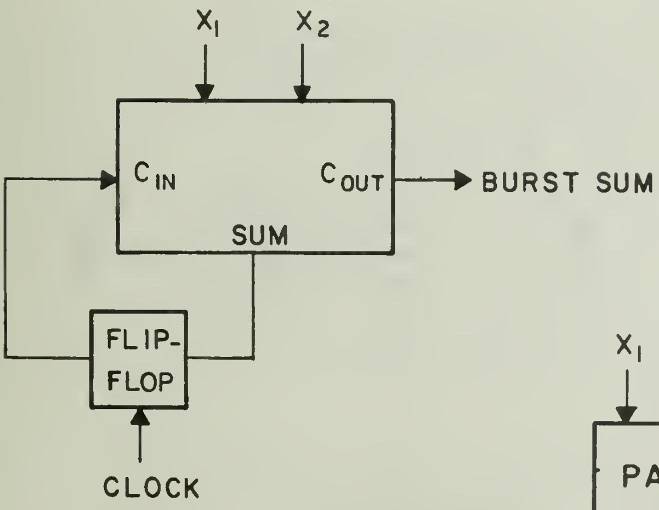
This PA circuit is all that is required to add two unsigned Bursts or two Bursts with biased Burst representation. To subtract biased Bursts an inverter is inserted in the input lead of the subtrahend. The output is always a properly scaled, uncompacted Burst. Uncompacted Bursts are completely acceptable as input.

2.3 Design and Operational Principles of a PA Multiplier

A multiplier has been designed in which the multiplier, multiplicand and the product are all in Burst format. It will be helpful to refer to Figure 5 during the discussion of the operation of the PA multiplier.

The multiplicand ("B" in Figure 5) flows through the shift register at the top of the diagram. The shift register is exactly the length of one Burst, and so, like a BSR, always contains as many ones as the value of the number being represented. Notice that the multiplier ("A" in Figure 5) is ANDed with a clock signal and the output of this AND gate is used to clock the Perverted Adders. The top row of PAs will produce a scaled sum of the contents of the shift register. This sum will appear in paral-





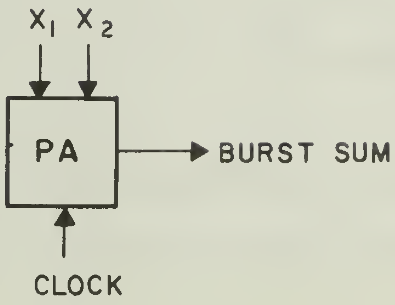


Figure 4. Logic diagram of a Full Adder (top), Full Adder and Flip-flop connected as a PA (lower left), Symbol for PA (lower right).

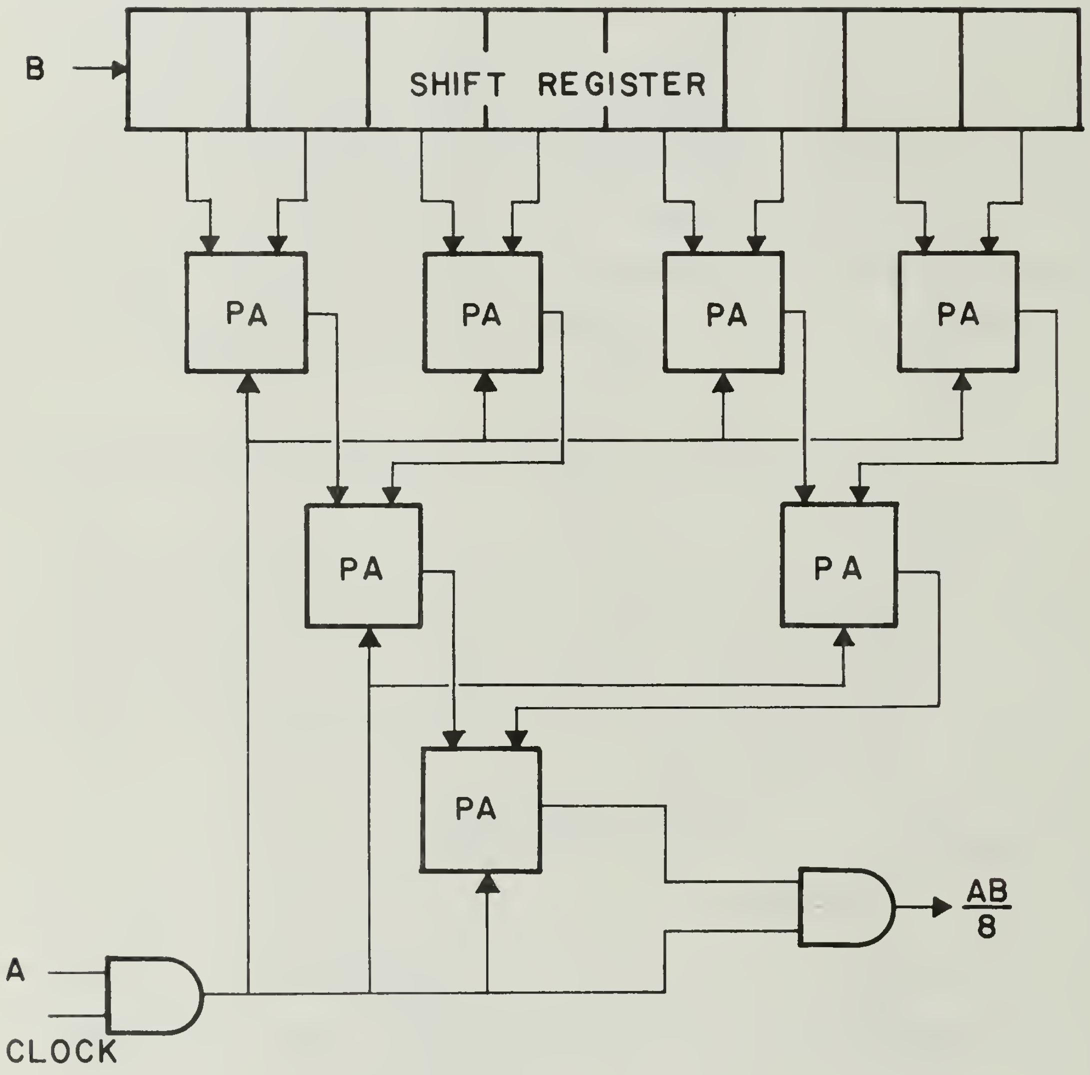


Figure 5. Circuit for the PA multiplier.

lel on the four outputs from the top row. These four lines are inputs to the second row of PAs and the outputs of these go to the inputs of the final PA. The actual addition takes place only when a one is present on A to produce the clocking signal. The result is that B will be added into the PA tree A times and the output will be the scaled sum of these additions. Each level of the PA tree causes a scaling by a factor of 2, hence the factor of 8 for the multiplier in Figure 5.

The delay through this circuit is dependent on the Burst sum propogation. Recalling that this is actually the carry output as the full adder is normally used and that that output is often faster than the production of the sum, propagation speed through the tree is quite good. Speeds of 5 to 8 MHz are possible with conventional TTL ICs.

2.4 Design and Operational Principles of a PA Divider

A PA divider can be constructed by adding only a few components to the multiplier. Recall from the discussion of the PA multiplier that the PA tree produces one output pulse (on the average) for every 8 ones added into the top row of PAs. If one were to "load" the tree with 8-n ones, then just n more ones would produce a one at the output. This would, in effect, be performing division by n on the input stream supplying the additional pulses. If every time the output of the tree produces a one, we load the tree again, we are performing a continuous division by n. The number of ones needed to load the tree (8-n) is easily generated by inverting the divisor and passing it into a shift register. An example should serve to clarify this division algorithm. Assume we wish to calculate A/B in an 8 bit-per-Burst system. A can be any number, B will be assumed to be 3. Passing B through an inverter produces Bursts with 5 ones. Begin by adding the 5 ones (in parallel) into the PA tree. Begin to add A into the tree, one bit at a time, into one input to the top row of the PA tree. After three ones (on the average) from A are added into the tree, a one will be output. At this time we again add 5 ones (the inverted B) into the tree and proceed adding more bits from A.

A divider of this type needs a two phase clock. During one clock phase the inverted divisor is added into the tree (when needed), and during the other phase the dividend is added into one input of one PA in the top row. The PA tree must be clocked on both phases.

It is possible, at the expense of some hardware to build a divider using a single phase clock. This divider runs at speeds comparable to the multiplier as it is necessary to wait for only one ripple through the tree each cycle instead of two. This can be performed in either of two ways. The inverted divisor can be staticized (i.e., held without shifting for 8 clock cycles) in a register in such a way that the one bits in that register do not interfere with the input of the dividend. It may also be accomplished by moving the one bits (specifically the one that falls on the input lead reserved for the dividend) as necessary with combinational logic. This latter approach is used in the circuits in Chapter 3 and is illustrated in Figure 6. The combinational logic shown in Figure 6 implements the following equations:

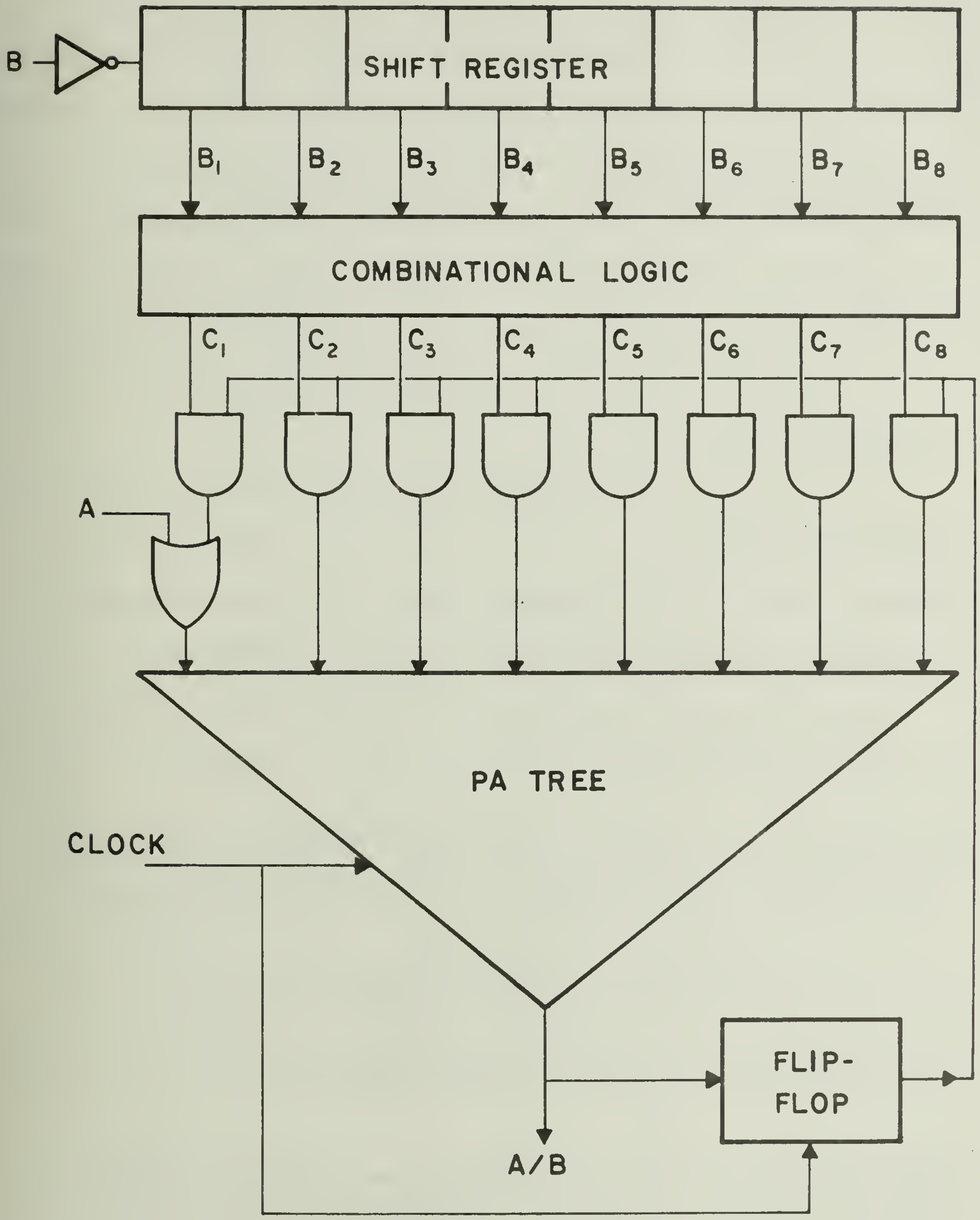


Figure 6. Circuit for the PA divider.

$$C_1 = B_1\ B_2\ B_3\ B_4\ B_5\ B_6\ B_7\ B_8 \qquad (3)$$

$$C_2 = B_1 + B_2 \qquad (4)$$

$$C_3 = B_1\ B_2 + B_3 \qquad (5)$$

$$C_4 = B_1\ B_2\ B_3 + B_4 \qquad (6)$$

$$C_5 = B_1\ B_2\ B_3\ B_4 + B_5 \qquad (7)$$

$$C_6 = B_1\ B_2\ B_3\ B_4\ B_5 + B_6 \qquad (8)$$

$$C_7 = B_1\ B_2\ B_3\ B_4\ B_5\ B_6 + B_7 \qquad (9)$$

$$C_8 = B_1\ B_2\ B_3\ B_4\ B_5\ B_6\ B_7 + B_8 \qquad (10)$$

The quotient in this divider is not scaled, therefore overflow occurs only upon division by numbers smaller than 1. Division of any number by zero yields Bursts completely filled with ones.

2.5 Use of PA Circuits in Signed Arithmetic Processors

Addition and subtraction of biased Bursts were covered in section 2.2 and proved to be trivial to implement. Multiplication and division of sign and magnitude Bursts is also very simple. The only additional circuitry needed is an exclusive or gate to process the sign information. The implementation of a sign and magnitude adder or subtractor and of a biased PA multiplier are described in this section. The most economical way to perform division on biased Bursts appears to involve conversion to sign and magnitude and back again. In each of these circuits it is necessary to staticize Bursts in registers. In each case the data is delayed for a total of only one or two Burst periods.

2.5.1 Sign and Magnitude PA Adder/Subtractor

A possible design for a Sign & Magnitude (S&M) PA Adder is shown in Figure 7. The principle components are two PAs and an n+1 bit bi-directional shift register. Not shown are an n bit parallel load shift register and the clocking circuitry to gate the result from one register to the other and to clear the bi-directional shift register after every Burst. This design is capable of accepting compacted or uncompacted Bursts and transmits compacted Bursts.

The 4 inputs to this adder (2 Bursts and two signs) are duplicated above each of the PAs in Figure 7. The gating between the input and the PAs causes the left PA to add only positive numbers, the PA on the right adds only negative numbers. The outputs of the PAs are used to control the shifting of the bi-directional shift register. The shift register is end connected via inverters. This ensures that the number of ones in the register corresponds to the magnitude regardless of the arrival times of positive or negative addend or augend. The left most bit of the re-

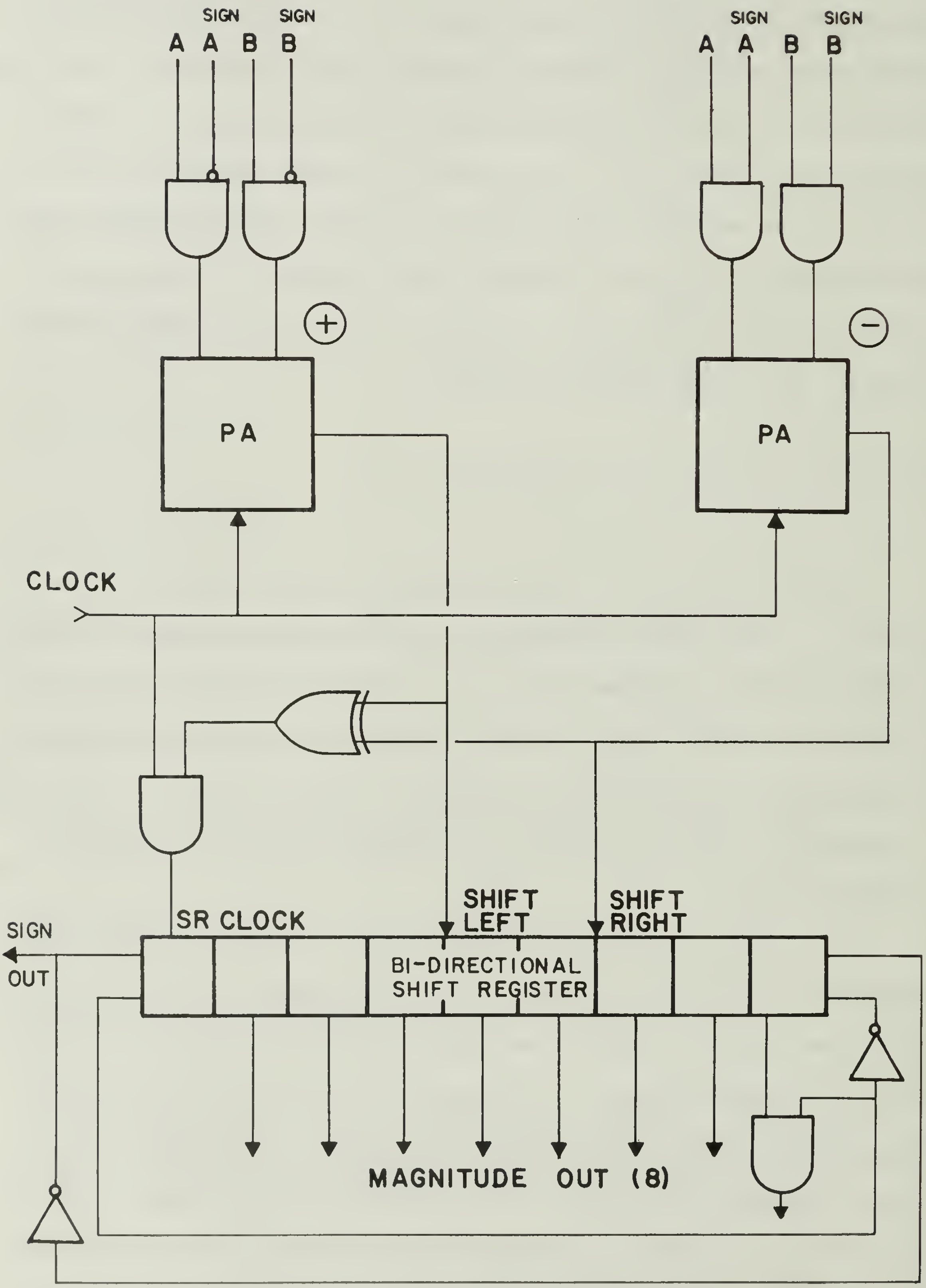


Figure 7. A design for a sign and magnitude adder.

gister always contains the sign bit according to the convention of a one for negative. The eight output bits are gated out in parallel, one of these bits being generated by the OR of the two end bits.

2.5.2 Biased Representation PA Multiplier

By adding additional circuitry to the PA multiplier of Figure 5 it is possible to multiply biased Bursts directly with PAs. A mathematical explanation of the principles of operation of this circuit will simplify understanding its functioning.

A biased Burst representation of n time slots per Burst will contain the number of ones specified by the following equations:

$$A = a + n/2 \qquad (11)$$

$$B = b + n/2 \qquad (12)$$

The capital letters represent the number of "1" bits in the Burst. The lower case letters are the numeric value represented and n/2 is the biasing term.

The product of two biased Bursts should then be

$$\frac{2ab}{n} + n/2 \qquad (13)$$

Where "a" and "b" have been multiplied, scaled and biased. In terms of the number of bits in the incoming Bursts the multiplication function becomes:

$$\frac{2AB}{n} - A - B + n. \tag{14}$$

The PA multiplier of Figure 5 performs the AB/n part of this function. Remembering that an output pulse from the PA multiplier is possible only when A has a "1", the function then becomes

$$\frac{AB}{n} - A^* - B + n \tag{15}$$

where A* represents the A pulses that do not produce a product pulse from the PA multiplier. Looking farther we see that -B + n is just $\overline{B}$, so we have

$$\frac{AB}{n} - A^* + \overline{B} \tag{16}$$

This function is realized by the circuit in Figure 8. Not shown in Figure 8 are a register to collect $\overline{B}$ left packed and a register to accept and serially transmit the data as well as the necessary clocking circuitry.

2.5.3 Representation Converters

No biased representation divider has been designed. The only reasonable approach would be to convert to sign and magnitude and back again. Representation converters are not very complex, but will involve running the bit rate for the biased representation at twice that of the sign and magnitude

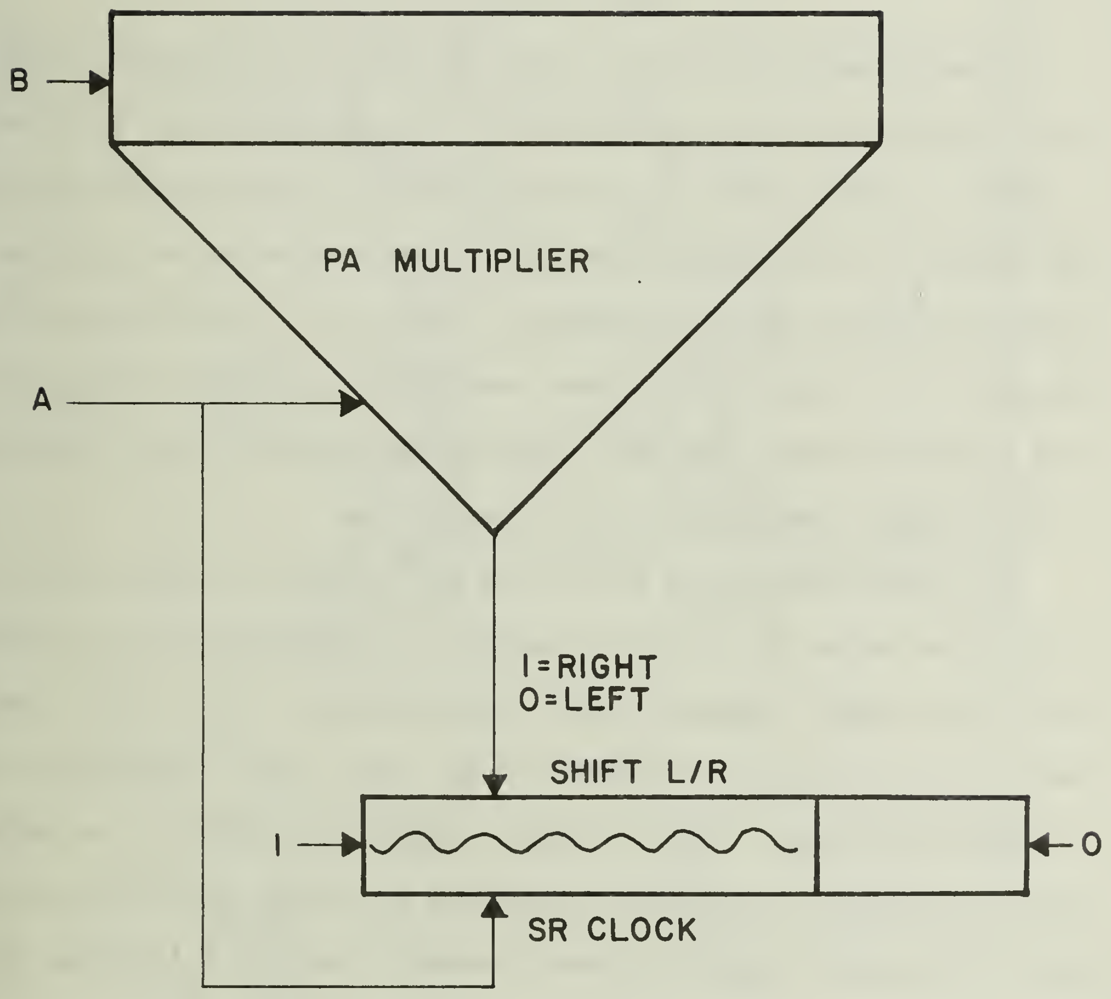


Figure 8. A multiplier for biased Bursts.

representation in order to prevent loss of information. A sign and magnitude to biased converter is shown in Figure 9 and a converter from biased to sign and magnitude in Figure 10.

2.6 Modified PA Trees

Several changes to the basic PA tree design are possible to adapt the circuit better to certain applications. If lower speeds can be tolerated in order to save hardware the circuit principle illustrated in Figure 11 can be used. An additional advantage is that the random noise caused by stored bits in the tree is decreased. This circuit uses only one flip-flop per level in the tree. This scheme requires carry ripples through the tree to proceed across and down. The optimum method of carry handling is to ripple toward the center of the tree and down.

All PA trees described so far have scaled by an integer power of 2. It is possible to use this type of tree with noninteger power of two length Bursts, but dynamic range of number representation is lost. For example, ten bit Bursts scaled by a normal 4 level tree, cause a reduction of 37% of the dynamic range. Two such stages cause a loss of 61%. The method shown in Figure 12 illustrates a technique to adjust the scaling to precisely the value needed to preserve dynamic range. Figure 12 illustrates the technique for 10 bit Bursts but the generalization to any number is straight forward.

To modify the tree to scale by the desired factor, the smallest tree is constructed which is capable of handling the desired number of inputs. The

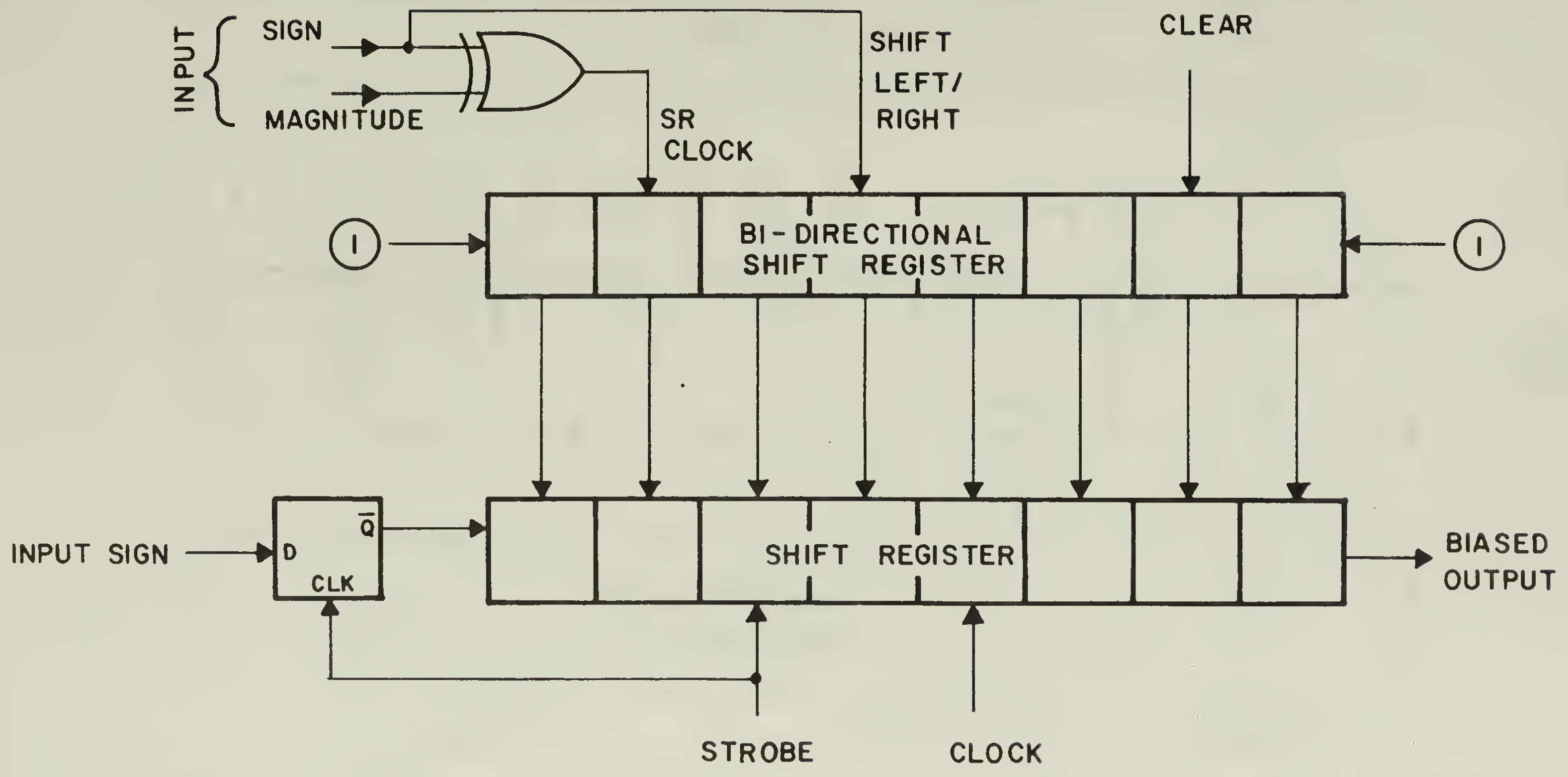


Figure 9. A converter from sign and magnitude to biased Bursts.

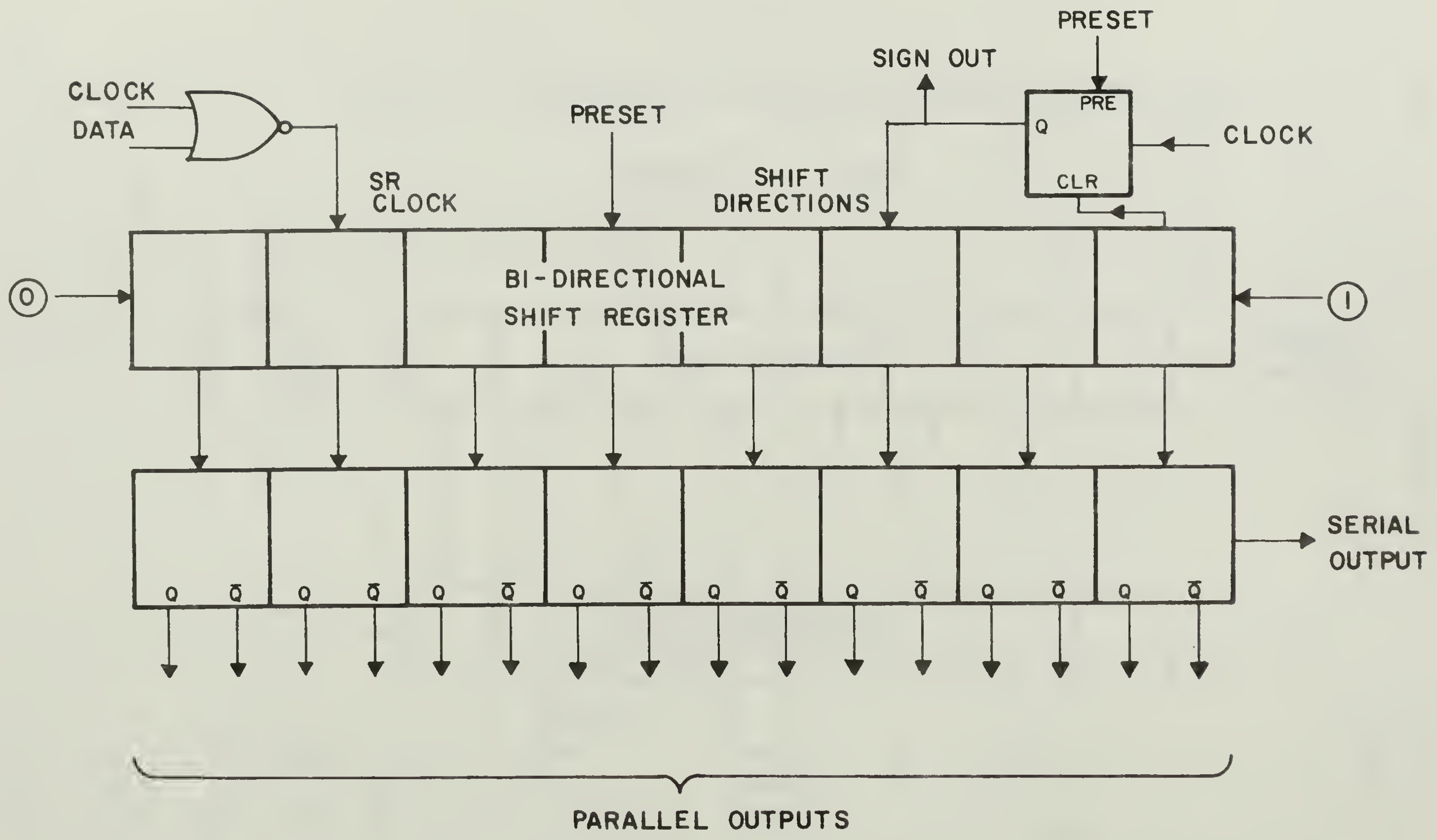


Figure 10. A converter from biased Bursts to sign and magnitude representation. The upper register is initialized to ones; when the right bit becomes zero the sign flip-flop is cleared. When the result is gated to the lower register it may be accessed serially or in parallel. The $\overline{Q}$ outputs can be used in the PA divider and eliminate the need for combinational logic for shifting.

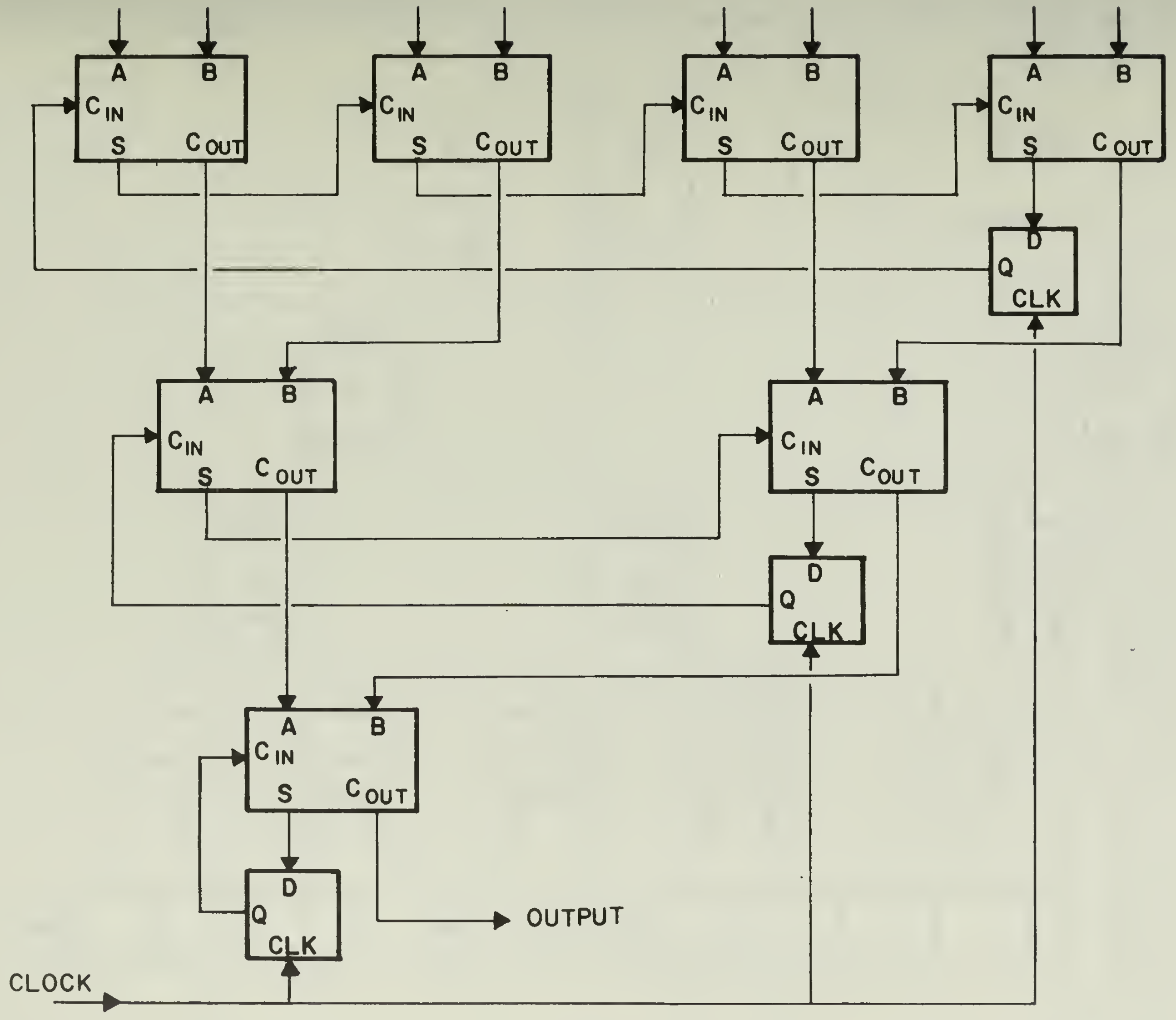


Figure 11. A PA tree design using less hardware. Some of the lost speed may be regained by placing the flip-flop in the center of each row to shorten the distance the ripple carries must travel.

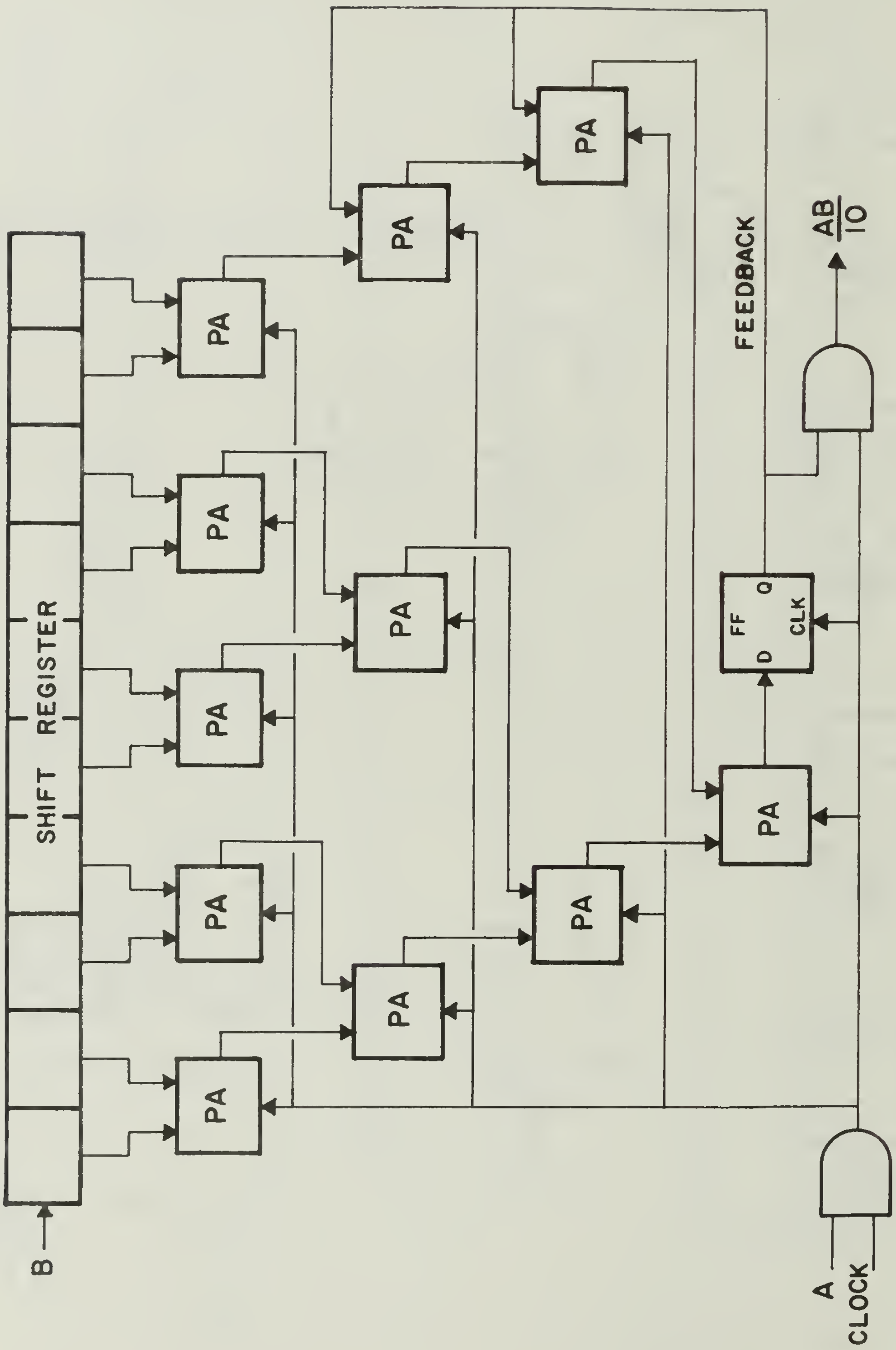


Figure 12. Feedback techniques used to create a PA tree that scales results by 10. Scaling by any integer is possible.

unused inputs (regardless of their position in the tree) are feedback points to preload the tree in a manner similar to that described in section 2.4 for the PA divider.

Table 2 shows various comparisons for performance for the techniques presented in this section for various Burst lengths. Hardware complexity and delay times are given for normal PA trees as well as non-power-of-two trees and for both techniques of 1 storage element per level.

Table 2: Comparisons of PA Trees

	Bits per Burst				
	8	10	12	16	20
Hardware Complexity					
Full Adders	7	11	12	15	21
Flip Flops					
Normal	7	11	12	15	21
Modified	3	4	4	4	5
Delays (ripples)					
Normal	3	4	4	4	5
Carry to End	6	8	9	11	14
Carry to Center	5	6	7	8	10

2.7 Multi-input PA Adders

Burst addition as defined with automatic scaling is not associative; i.e.

$$(A + B) + C \neq A + (B + C) \quad (17)$$

After calculation the left side will be

$$\frac{A}{4} + \frac{B}{4} + \frac{C}{2} \quad (18)$$

The right side will be

$$\frac{A}{2} + \frac{B}{4} + \frac{C}{4} \quad (19)$$

If it is desired to generate

$$\frac{A}{3} + \frac{B}{3} + \frac{C}{3} \quad (20)$$

this can be accomplished by a PA tree with feedback. Using a four input tree A,B and C are connected to three inputs and feedback is supplied to the fourth.

This scheme can be generalized to any number of inputs.

3 A BURST CPU TO DEMONSTRATE PAs

A Burst CPU capable of multiplying or dividing two 8-bit Bursts has been constructed. The input Bursts have each bit position determined by a front panel switch. The output is displayed on a pair of seven segment LED readouts. The counter which drives the readouts is selectable between 8 and 64 clock cycles for the display. The clock may be external or internally derived by dividing the powerline frequency by 1, 2, 4, 8, or 16. The remainder of this chapter is devoted to a more detailed description of this CPU. Figure 13 is a photograph of this CPU which is complete on two circuit boards plus the front panel components. Figure 14 is a block diagram of the CPU and should be referenced to determine how the blocks of circuitry to be discussed relate to each other. Consult Appendices A and B for lists of ICs and board pin connections respectively.

3.1 Clock and Control Circuits

The clock and control signal generation circuits occupy part of board 1 and can be seen in the schematic diagram at Figure 15. A.C. ripple from the power supply is directed to a one-shot multivibrator. The resulting TTL compatible signal is fed to the selector switch and to a 74163 4-bit binary counter, the outputs of which are additional clock signals which may be selected by S1. The external clock input is also shown. This clock signal is used in the Burst

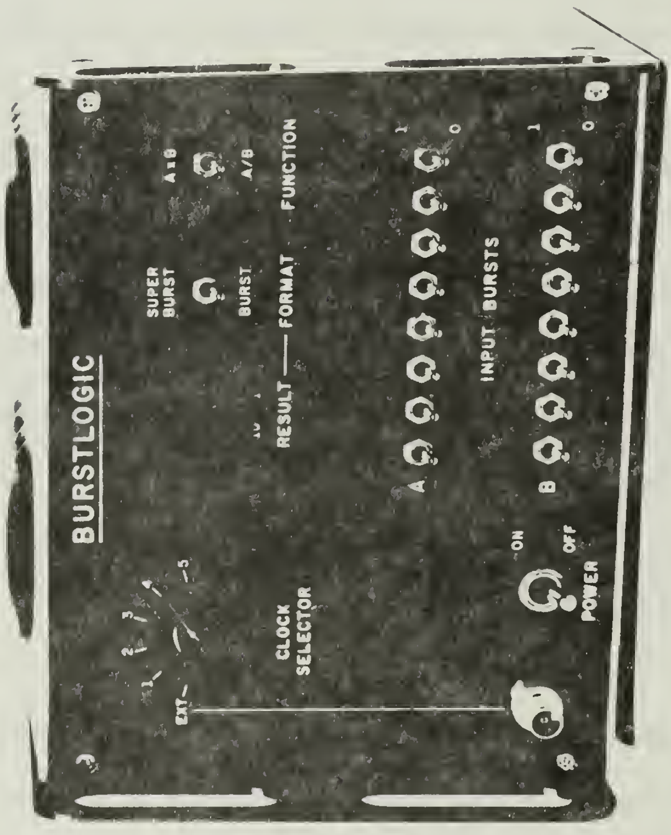


Figure 13. Photograph of the BURSTLOGIC CPU.

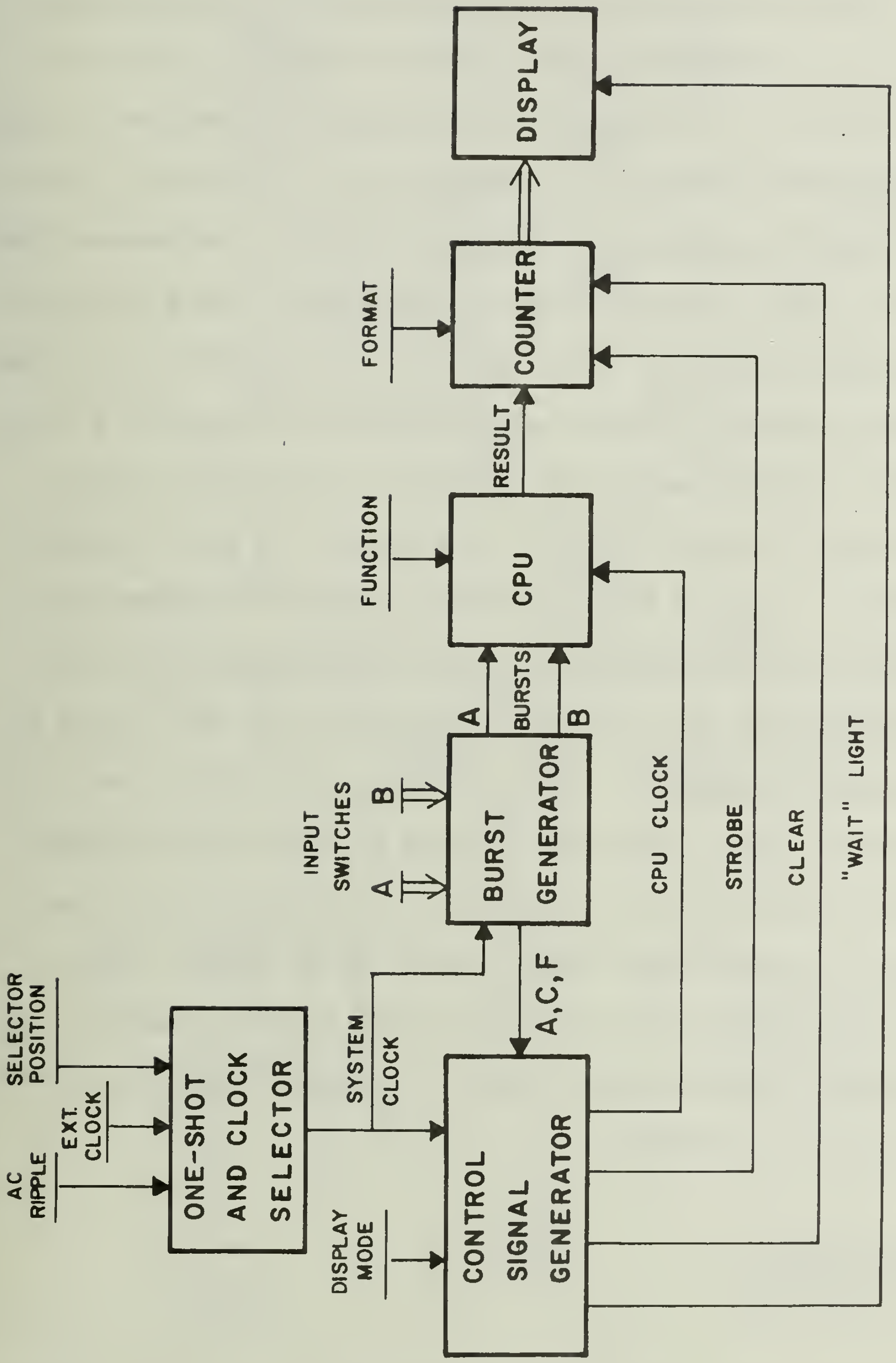


Figure 14. Block diagram of the BURSTLOGIC demonstration CPU.

generator, (see Figure 17) as well as in one of the NORs of IC 16.

One of the by-products of Burst generation is the signal labeled "A", shown as an input to the clock section in Figure 15. The timing diagram of Figure 16, illustrates the relationship between the selected clock (system clock on Figure 16) and signal "A". This signal is used to drive a 7473 flip-flop and a 74163 4-bit counter. The feedback from Q_D to the CLR input causes the 74163 to divide by 9. These two signals are used to gate the counter for display of 8 or 64 clock cycles respectively. The selection is made by either grounding or applying 5 volts to the display select input. This is done by a front panel switch.

The output of the AOI (IC 20) is the same as the signal labeled "wait light" on Figure 16 for the selection of the signal from the 7473. If the 74163 was selected the signal would remain low for 64 clock cycles and high for 8 cycles. The output of the AOI is used to derive the remaining control signals. NORing it with the system clock produces the CPU clock. This signal is out of phase with the system clock and is inhibited when the display is in its update cycle (see Figure 16). "C" and "F" are signals similar to "A" but of different phases. In conjunction with the output of the AOI these generate the STROBE and CLEAR signals for the counter. The "wait light" is driven by an open collector inverter.

3.2 Burst Generators

The input Bursts to the CPU are generated by the circuit of Figure 17. The heart of the Burst generator is the shift register and the NAND

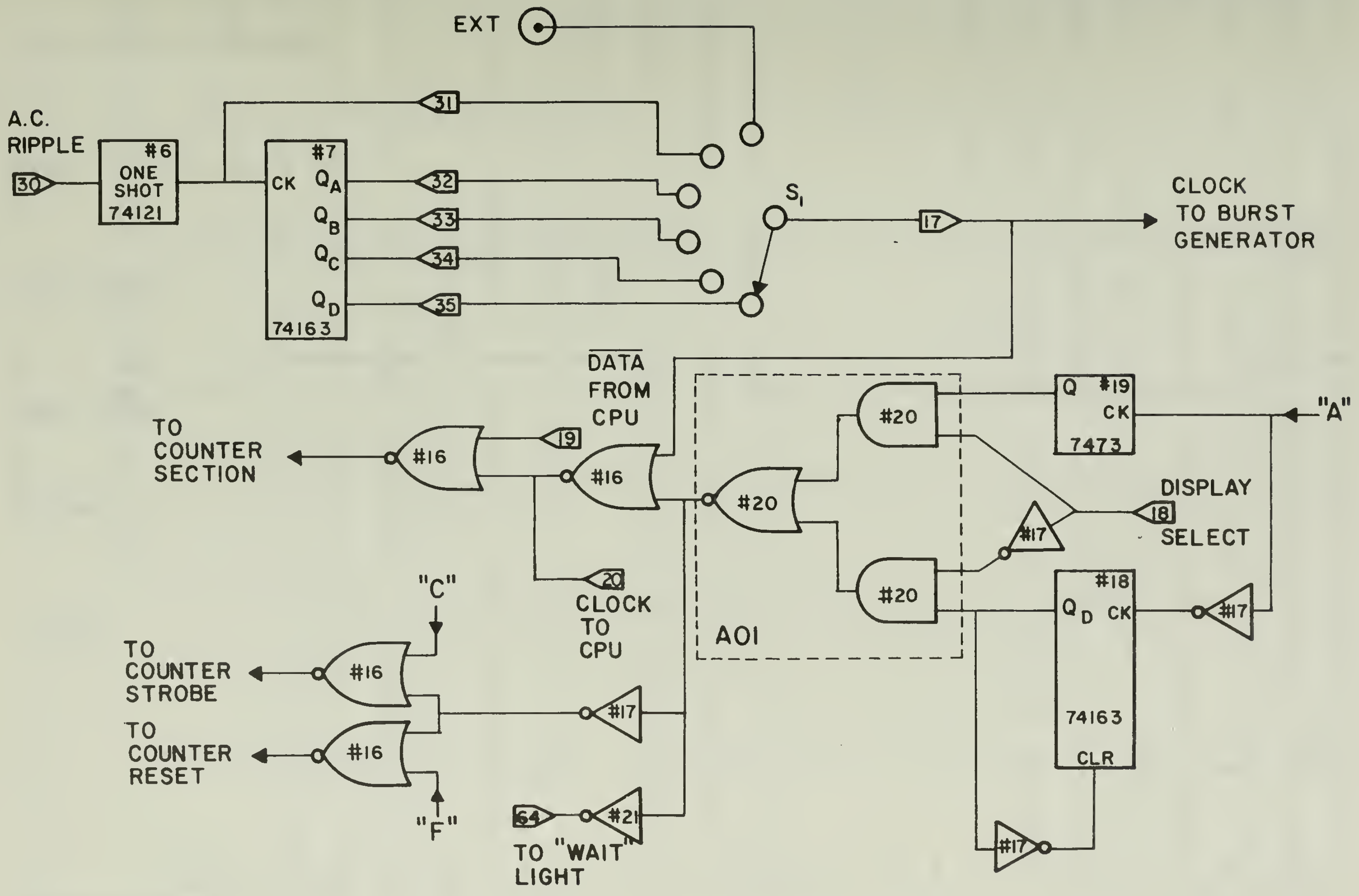


Figure 15. Schematic of the clock and control signal generation circuitry of BURSTLOGIC. The ICs are on Board 1.

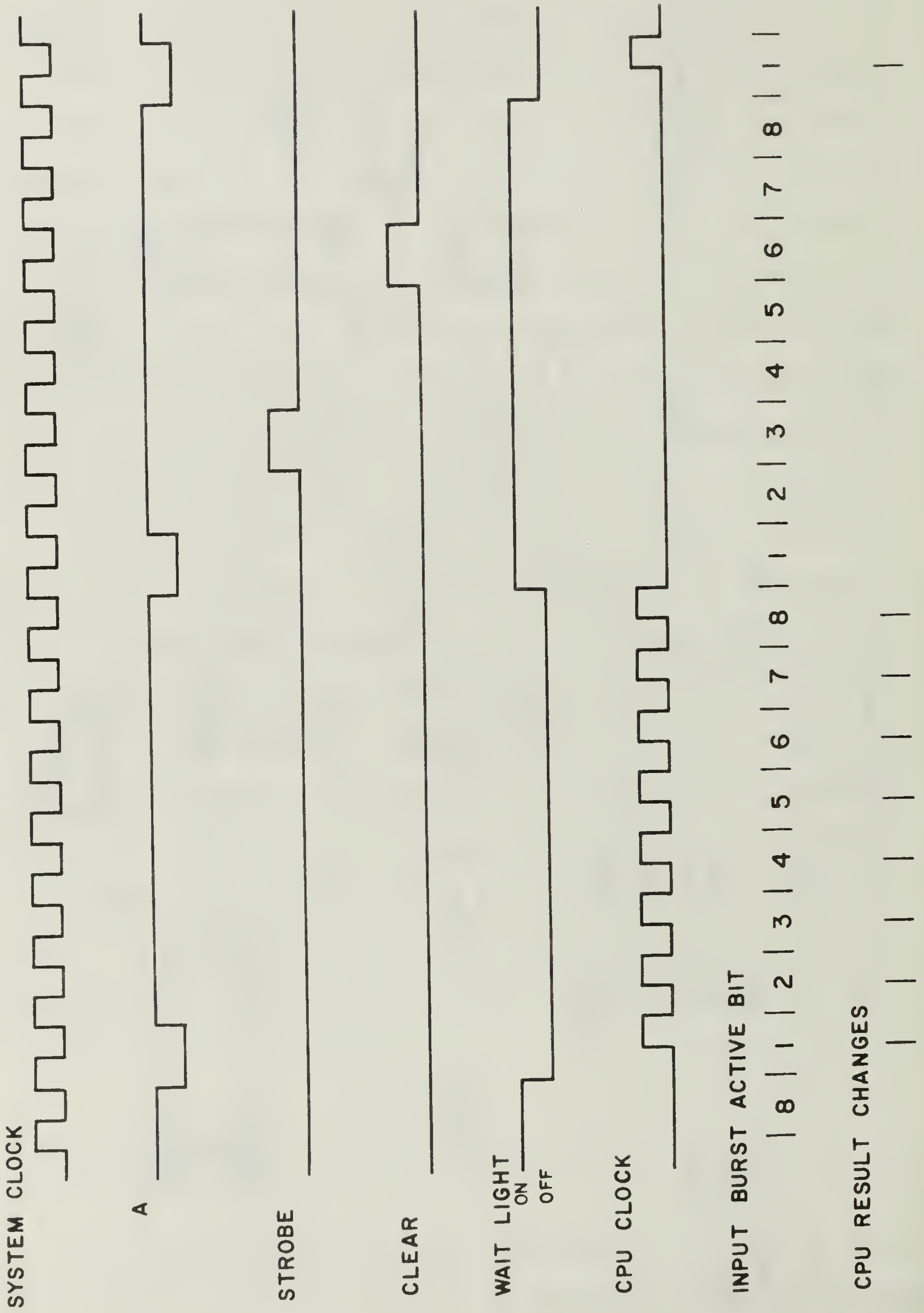


Figure 16. Timing diagram for the BURSTLOGIC CPU.

gate in the center of the diagram. These are connected to cause the shift register to always contain one zero and seven ones. The circuit is self-starting in less than 8 cycles. The outputs of Q_A, Q_C and Q_F are used by the clock and control circuits. Using the eight signals Q_A to Q_H any number of Burst generators can be driven. Each Burst generator (both are shown) consists of 8 switches, 8 2-input OR gates an 8-input NAND and a D flip-flop. These Burst generators generate uncompacted Bursts.

3.3 The CPU

The actual arithmetic functions of multiplication and division are performed by the circuitry on board 2. This will be illustrated and discussed in two parts: 1) the shift register and combinational logic section and 2) the PA tree.

The design is basically the PA divider discussed in section 2.4. This has been augmented with a few gates to change easily between the two functions. The switching is performed by a front panel switch which supplies either 5 volts or ground to the gates. During multiplication the circuit is operated with the divider's combinational logic in place and functioning. This logic is two levels deep and does not seriously degrade the multiplier.

3.3.1 Shift Register and Combinational Logic

The CPU clock signal shown in Figure 17 is used to clock the shift

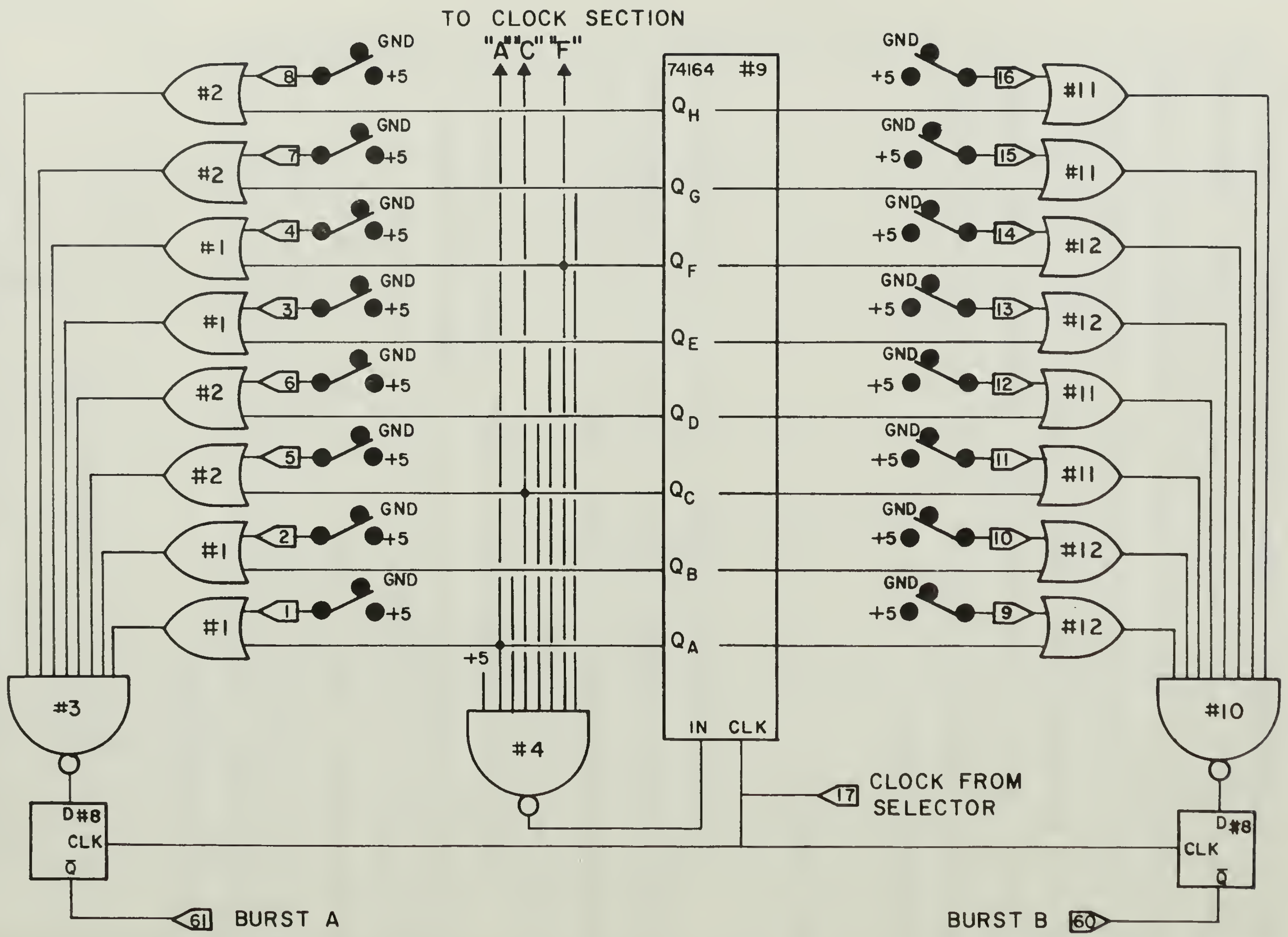


Figure 17. Circuitry to generate uncompacted Bursts for the BURSTLOGIC CPU. The ICs are on Board 1.

register through which the B Burst will flow (see Figure 18). This Burst enters the register through an XOR gate which serves to pass or to invert the B Burst depending on the mode select input (inversion is necessary for division). Other functions of the mode select are to inhibit the A Burst from being fed directly into the first PA's input, and to inhibit feedback from the output of the PA tree.

To clarify the diagram of Figure 18 connections between the shift register and the combinational logic are lettered. The use of these letters is local to this one diagram and is unrelated to similar letters in the Burst generator for example. Signals L & M are identical. Two gates were used to prevent the fanout problems of driving 14 inputs in the combinational logic section. The outputs of the combinational logic are labeled C_1 to C_8. The corresponding inputs are labeled similarly on the PA tree diagram (Figure 19).

3.3.2 PA Tree

The PA tree occupies the second half of board 2. Complete PAs are used rather than the circuits with less than one flip-flop per full adder shown in section 2.6.

The mode select permits the flip-flops to be clocked by the A Burst or by the CPU clock. The circuit description and operation are exactly those of the designs in Chapter 2. The final flip-flop is needed to hold the last result for feedback and to accurately toggle the counter. Note that

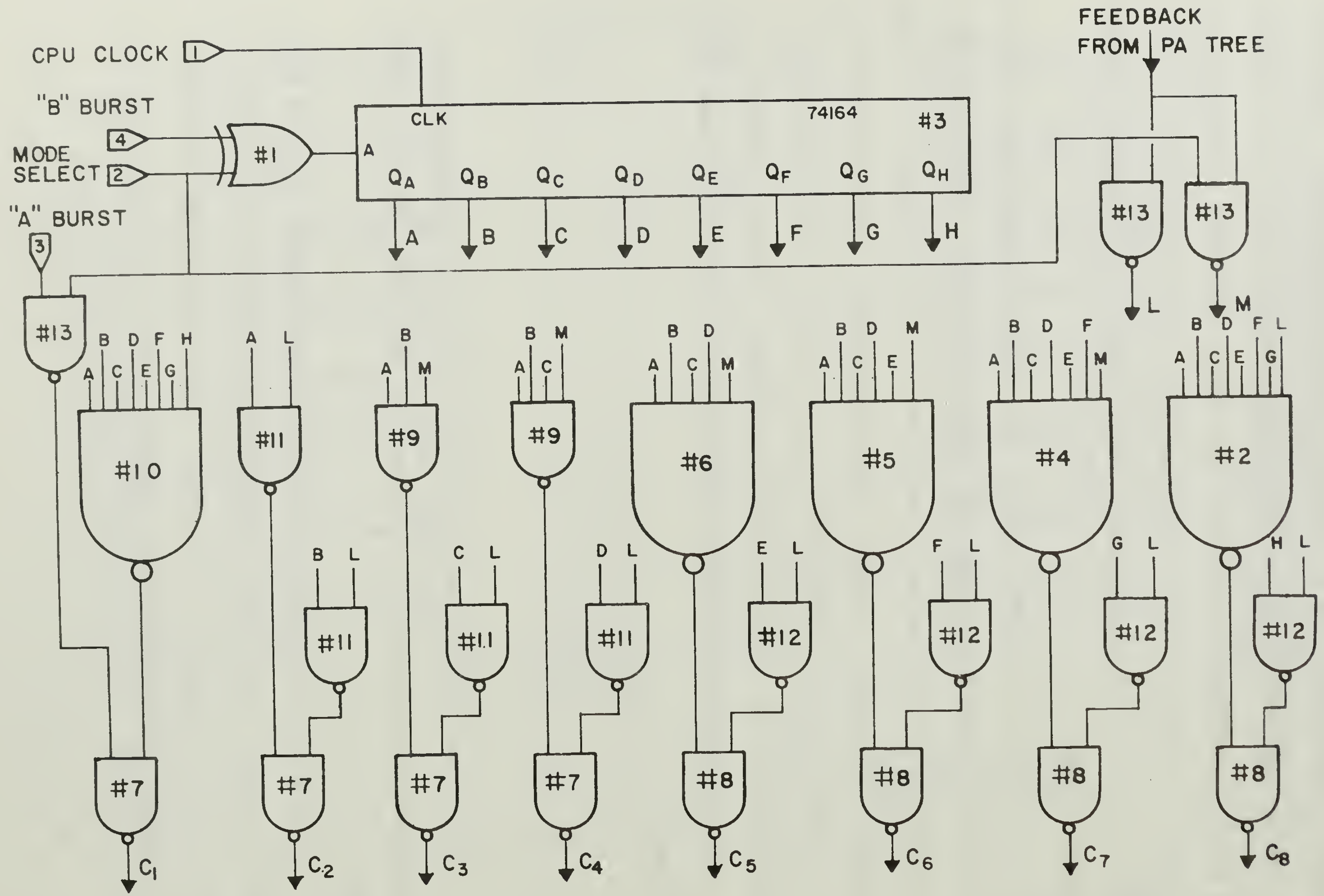


Figure 18. The shift register and combinational logic section of the BURSTLOGIC CPU. The ICs are on Board 2.

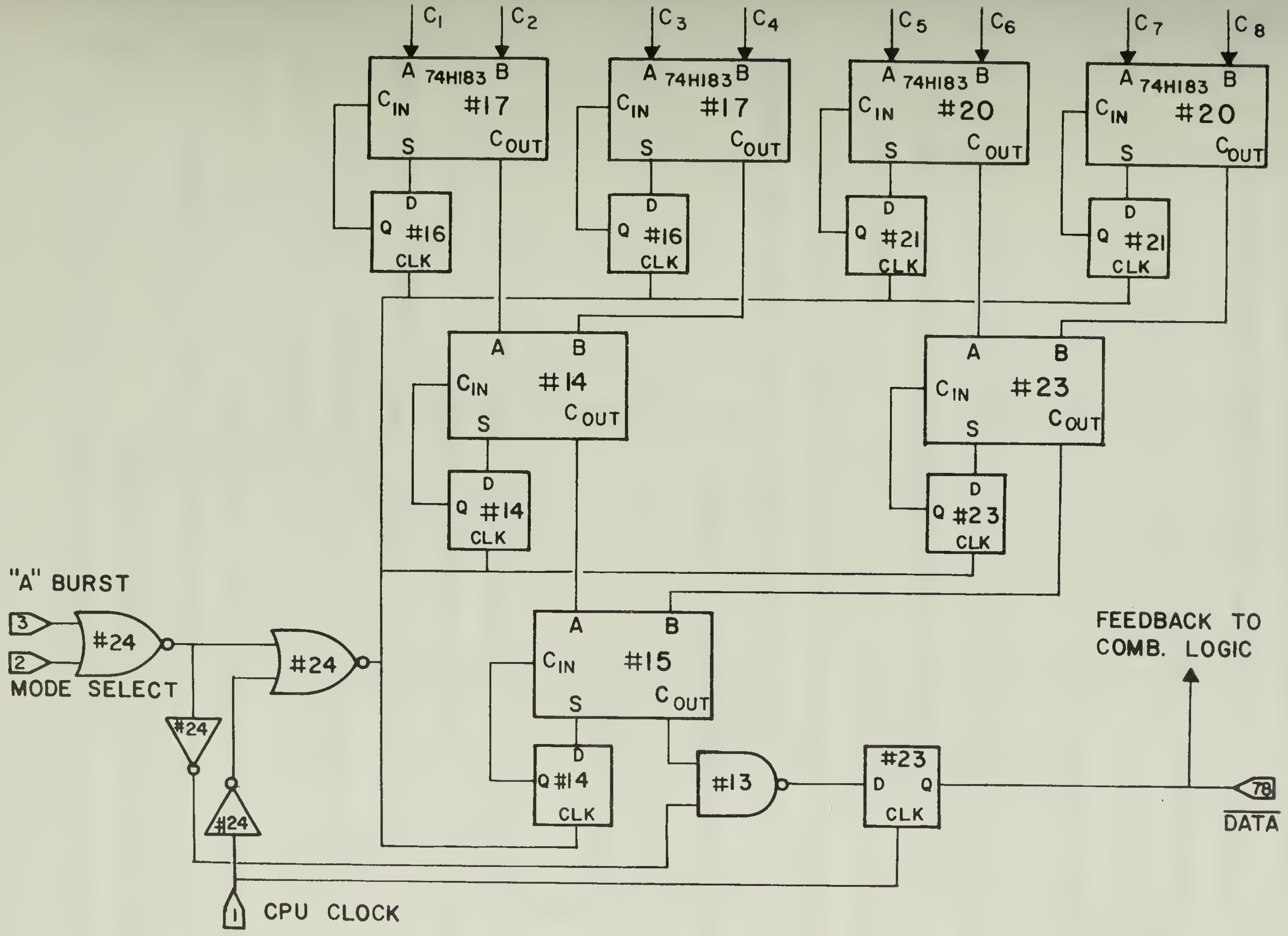


Figure 19. The PA tree section of BURSTLOGIC. Note that the result is inverted. The ICs are on Board 2.

the output signal is actually an inverted Burst signal. A NOR gate, shown on Figure 15, generates the necessary transitions from a high level to low level to toggle the counter whenever the output result is "1" and the CPU clock goes high.

3.4 Display Circuitry

The display section has three inputs. Besides the toggle signal mentioned above, the Strobe and Reset signals generated by the clock section are used to transfer data from the two digit decimal counter to the storage registers and to clear the counter. The result is converted to seven segment display format with the leading zero blanked. Figure 20 is a schematic of this section.

3.5 Discussion of Operation

The CPU will begin to run immediately when power is applied. An external TTL compatible clock must be furnished if the EXT position of the clock selector is chosen. Positions 1 to 5 correspond to internally generated clock speeds of 3.75, 7.5, 15, 30 and 60 Hz. The display format labeled "Burst" causes the unit to compute for 8 clock cycles and to drop into display update mode for the next 8 cycles. "Super Burst" will cause the unit to compute for 64 clock cycles and to drop into display update mode for the next 8 cycles.

A small dot in the seven segment display will be on whenever the

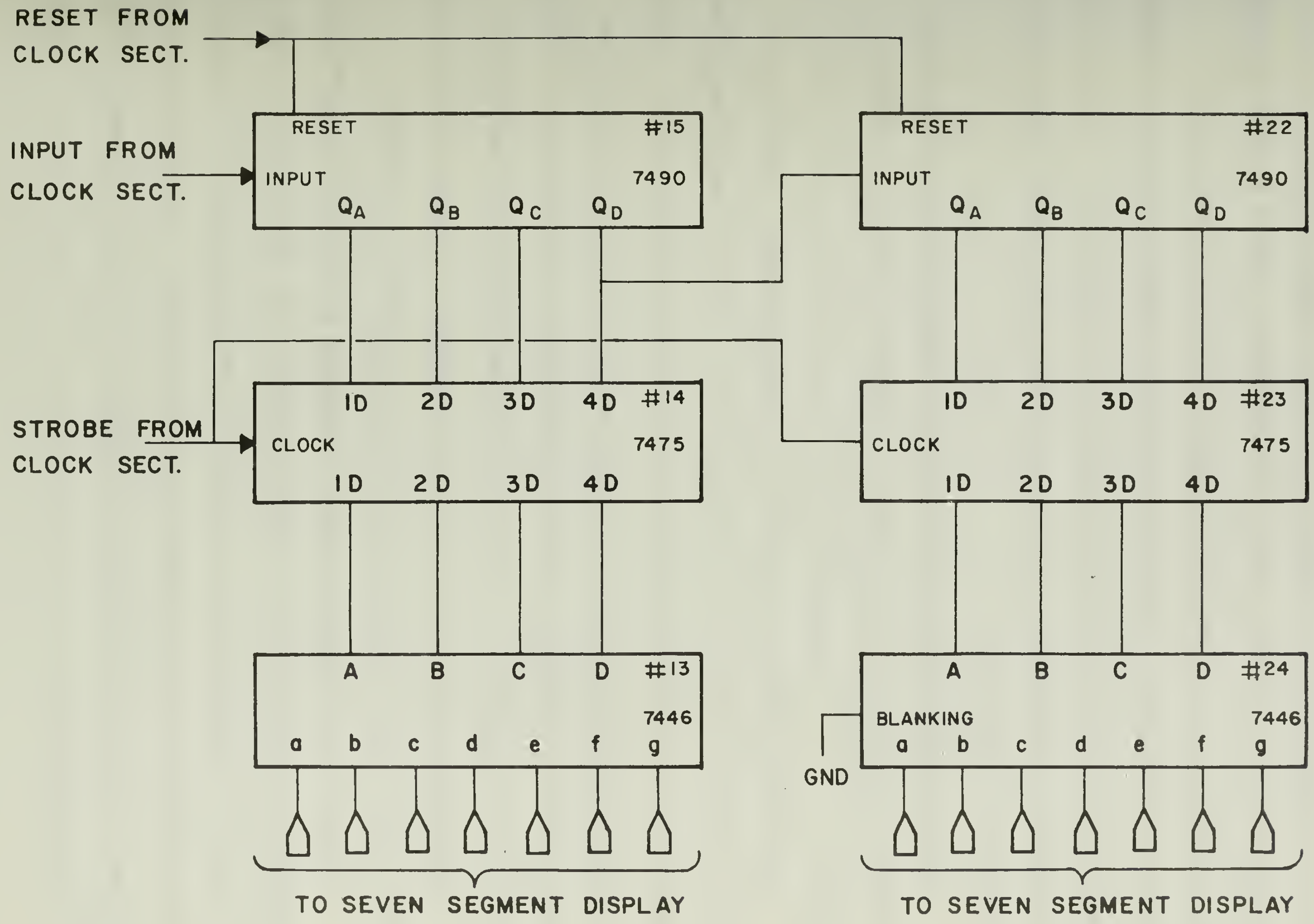


Figure 20. The display circuitry section of BURSTLOGIC. The ICs are on Board 1.

unit is performing display update. This "wait light", so called because the CPU must wait for the output to be performed, serves several purposes. When the output is not changing, it assures the user that the unit actually is computing. It is also useful when experimenting with fractional input Burst patterns, such as representing 3-1/2 by 3 4 3 4. The switches that control the input Bursts can be safely reset during the time that the light is on.

If these switches are changed at other times the results are unpredictable. The slower clock speeds are useful for this type of experimentation.

The display format of the unit is a decimal number, not octal as might be guessed from the 8-bit input Bursts. The interpretation of the result must also take into account the fact that multiplication is scaled to prevent overflow in the 8-bit output Burst. Division is not scaled, hence overflow occurs only upon devision by zero. Division by one yields precisely the numerator. Caution must be exercised when interpreting the display with fast external clocks as changing displays are easily misread because of the inability of the eye to perceive the changes.

Because it is inherent in the design of a PA tree to have some data stored and to not be able to define precisely when that data will come out, a given setting of the front panel switches may not always yield the same result (although the average will always be correct). It is possible, for example, to generate a value of 2-2/3 with 2 3 3 2 3 3 one time and with 2 2 4 2 2 4 the next.

4 CIRCUIT DESCRIPTIONS, EVALUATIONS AND COMPARISONS

The PA circuits described in chapter 2 have been presented as a viable method of performing arithmetic in low precision, real time applications. This chapter will present comparisons to other Burst designs and to the more conventional methods of weighted binary and the Digital Differential Analyzer (DDA). These comparisons will include design and operational descriptions and present data on hardware complexity/cost, aesthetics, speed of operation, noise immunity, and trade-offs between these and other design parameters.

4.1 BSR Circuits

The BSR was used in the ramp and vernier encoders presented in chapter 1. The operating principle of the BSR is that it is a current source or sink whose strength is proportional to the number of ones in the shift register. Addition is performed by current summing and re-encoding the result with a ramp or vernier encoder (see Figure 21). Two multiplier designs are possible. A simple design uses one BSR to control the voltage source of the second. This scheme is a complete multiplier when the output of the second BSR is re-encoded. If higher precision is needed and is available, as when the inputs come from a vernier encoder, a time slip cascade multiplier is possible. (1) This design requires an entire 10 block segment of input to be trapped and be available for processing. Selection of bits is made by the same

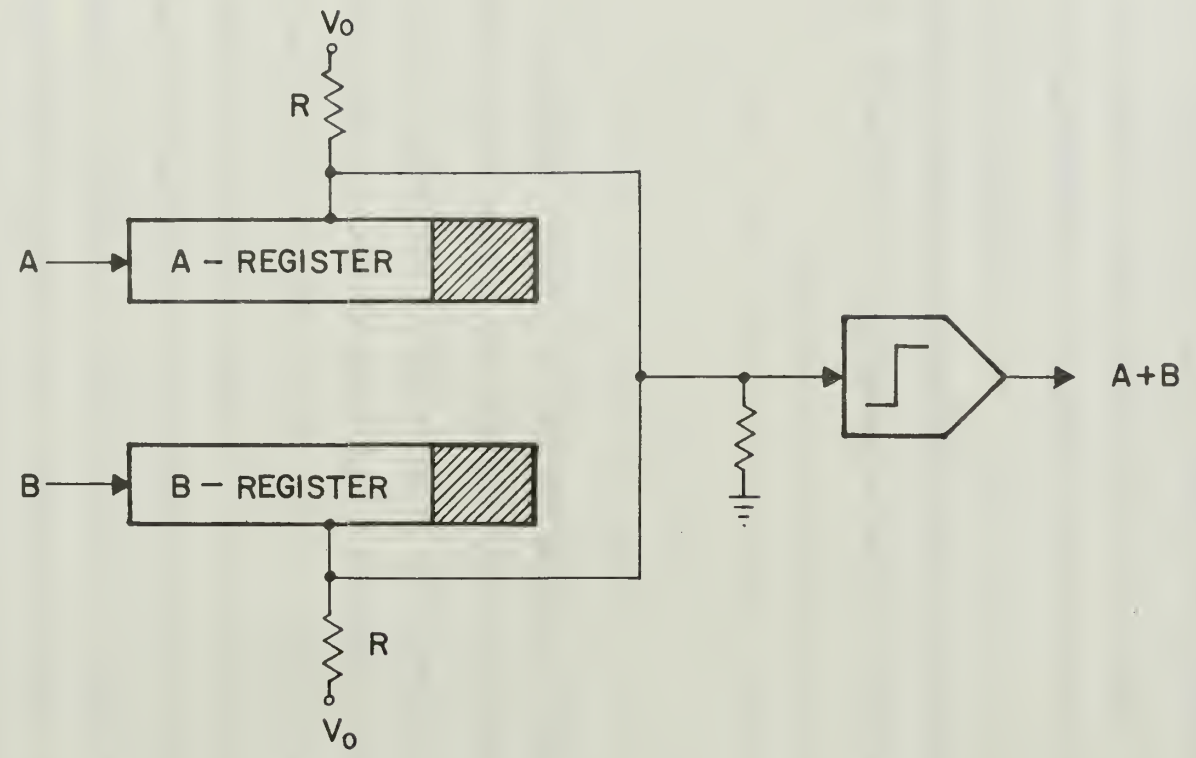


Figure 21. A BSR adder.

vernier encoder that is encoding the result.

The BSR approach is conceptually very simple, and has good possibilities for IC versions of entire Burst processors. The design of BSRs from discrete components is more difficult and is probably the reason for IEL projects using PA designs instead of BSRs.

BSR designs are quick to respond to input changes and have a universal building block (the BSR) much as PA designs do. The outputs are always compacted Bursts while the inputs may or may not be compacted. BSR designs are well adapted to designing digital filters. It is possible to vary the weights of the individual bits in the BSR so that the optimum n bit approximation to a given window (e.g. Hamming) is achieved. This can be done more accurately with BSRs than with any other method described in this chapter.

An approximation of the hardware complexity using 3 gate equivalents per flip-flop and 1 gate equivalent for each current source yields a complexity of about 160 gates for the adder and multiplier and 240 gates for the divider. As with all Burst methods there is no need for synchronization and noise immunity is high. Comparison charts are available in section 6 of this chapter.

4.2 PA Circuits

A thorough description of the PA arithmetic circuits will not be repeated here, but rather, unique characteristics and comparisons will be discussed. The comparisons will be made both on the basis of compacted and uncompacted Bursts; i.e., results will be given with and without a compactor running after the PA circuitry.

The PA adder is the simplest Burst adder with or without the compactor. The circuit is approximately 6 gate equivalents compared to 160 for BSRs. The complexity is about the same as the sampler adder (see 4.3.1), but that circuit has a significant drawback.

One problem with PA circuits, especially the multipliers and dividers is the storage of data in the PA tree. For circuits with complete PA, as in the CPU of chapter 3, 1/2 bit per level in the tree can be stored. In the tree with one flip-flop per level slightly less than one bit is stored. This storage can cause a somewhat higher deviation of the output from the expected value than BSR circuits. Computer simulations and experience with the CPU have shown that this deviation is a very complex function of the inputs and the condition of the tree prior to the operation. For applications which may be sensitive to such deviations longer averaging at the output completely eliminates this phenomenon at the expense of some high frequency response.

The complexity of the PA multiplier and divider are lower than any other Burst method (uncompacted Bursts) and for the low cost designs with one flip-flop per level, the complexity is the same as weighted binary of the same precision (16 bit PA tree compared to 4 bit binary), and is capable of comparable speeds.

4.3 Other Logic Design Approaches to Bursts

Logic design approaches to Burst arithmetic other than the PA

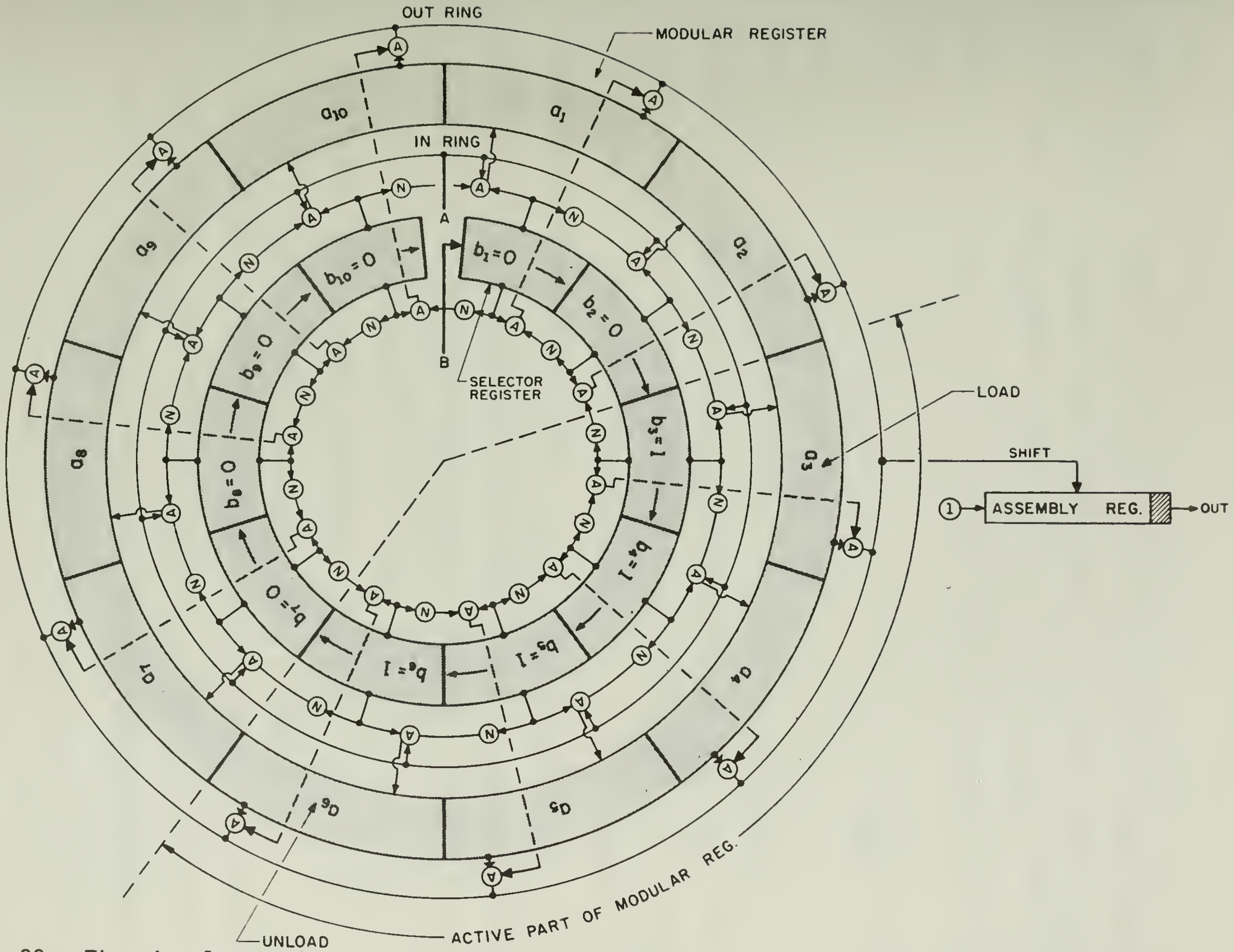


Figure 22. The circular registers and gating that forms the heart of the carousel multiplier/divider.

approach have been made. BURSTCALC has been constructed to demonstrate some of these ideas. The concept design for a low cost Burst CPU (LOCOBURST) was presented by Dr. Poppelbaum in March, 1976. Both of these approaches are now described.

4.3.1 BURSTCALC

BURSTCALC (5) contains a Burst adder, subtractor and multiplier/divider. The principle of operation of the adder is known as the selector adder. The two input Bursts are alternately sampled to determine the next output pulse. In an effort to eliminate problems of correlations between this sampling scheme and the input Bursts, the sampler changes phase every Burst. This scheme solves the problem for constant Bursts, but any scheme which discards data (this discards half) can have errors due to correlations between the discard pattern and the data.

The subtractor uses a shift register to obtain the value of one Burst. It then shifts in the opposite direction to perform the subtraction of the other Burst. Underflow is handled by changing the sign and shifting data back into the register after the sign change. If two such units are used and are one Burst length out of phase no data is lost.

The multiplier/divider uses the carousel register scheme shown in Figure 22. Multiplication requires a super Burst to complete the calculation. In each Burst of the multiplier a different bit position is checked. If the position so checked contains a one, the multiplicand is added into a reconstruction register. During one super Burst every bit position in the multiplier is checked, so the reconstruction registers will contain the product. This must be reformated to be transmitted as a super Burst. It is imperative that the inputs be compacted Bursts and that the amount of deviation of Burst sizes be minimal if correlation problems are again to be ignored. Note that nine out of ten data bits of the multiplier are discarded. This multiplier is also relatively slow to respond to changes in the inputs. An advantage which this multiplier shares with the PA multiplier is that most of the multiplier circuitry can be used in the divider.

The divider works on a system of comparing the length of the Burst in the divisor register with the length of the dividend Burst assembled in another register. When the two lengths are equal, the dividend is cleared and a one is added to the reconstruction register. The input Bursts must be compacted and the division should be quite constant. If uncompacted Bursts are satisfactory as output, the reconstruction circuitry can be eliminated. This divider responds quickly to changes of input if used without the reconstructor. Note that a noise pulse can confuse this divider as the leading and trailing edges of the Burst control the operation of the divider as they circulate in the registers.

4.3.2 LOCOBURST

The ideas and goals of LOCOBURST are very different from the other Burst designs discussed. It is intended to be a Burst CPU rather than just a single adder, etc. The heart of this design is the compactor/repeater shown in Figure 23. Half of this circuit is accepting bits and forming a compacted Burst while the other half is circulating and transmitting its contents. This same Burst will circulate and be transmitted until the other Burst is ready. The time for this to happen can vary from a Burst or two to ten or more depending on the function being performed and the numbers involved.

A very distinct advantage that this circuit has compared to the others is that it eliminates the persistant Burst problem of sum of products not equaling product of sums and sum of quotients not equaling quotient of sums. This is accomplished by performing an averaging before performing the arithmetic.

The addition scheme is essentially the same as the selector adder of BURSTCALC. The compactor/repeater is involved in the role of compacting these results. There are several clocking type signals involved, including the selector signal for the adder and a vernier type signal which is low for 9 clock cycles, then high for two cycles. Figure 24 shows the entire LOCOBURST arithmetic unit. The selector signal is labeled "s" and the vernier signal "e" in this figure.

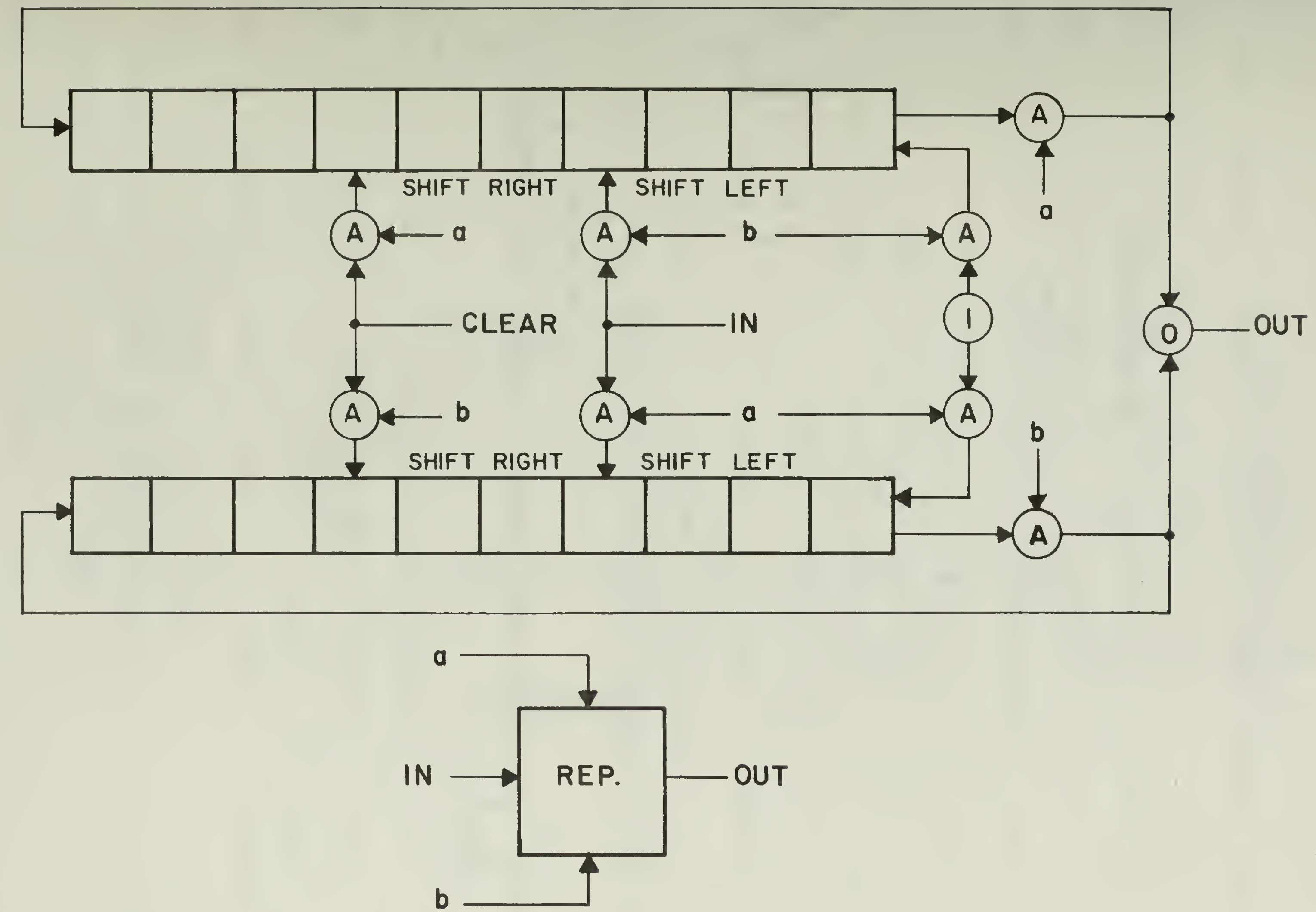


Figure 23. The compactor/repeater used in LOCOBURST.

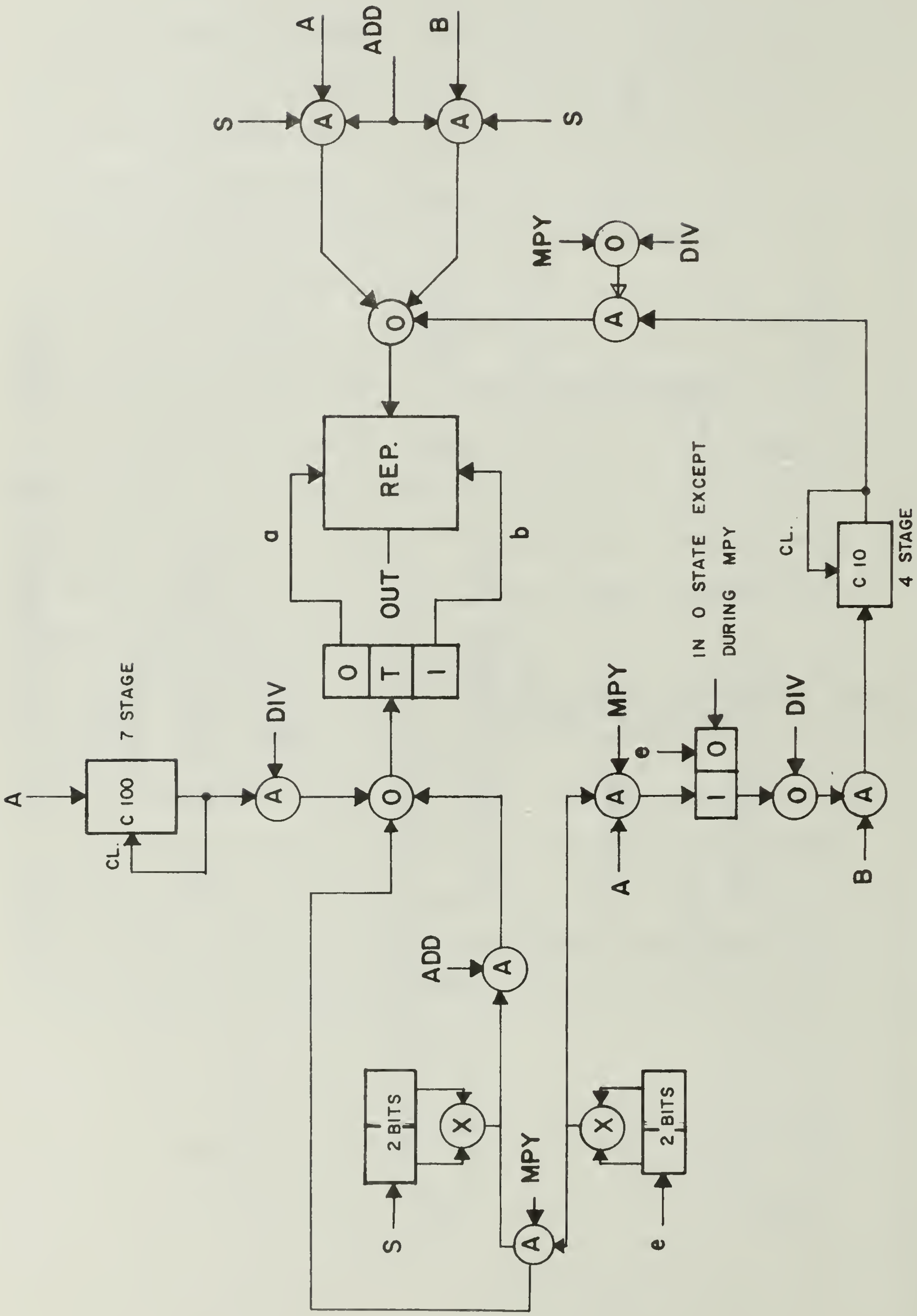


Figure 24. The LOCOBURST processor.

4.4 Weighted Binary

This comparison to weighted binary is included to show how Burst processing in general and the PA techniques in particular compare with more conventional techniques. Weighted binary is a much more compact representation than Bursts. It does, however, require synchronization in order to assign the proper weighting factor to each bit. It also has no inherent noise tolerance. Error correcting codes can improve noise tolerance, but these will not be discussed here.

The binary adder is, of course, very similar to the PA, and consists of a full adder and a flip-flop. Because of their similarity, the comparison of the PA to the binary adder is just a comparison of the two representations. The binary adder can process more data because of the more compact representation, but in the presence of noise the PA's answer will usually be more accurate.

The weighted binary multiplier used in these comparisons is shown in Figure 25. Binary multiplication will always yield more precision than contained in the input. This multiplier truncates the low order bits to make the output compatible with the input. No clocking nor control circuitry is shown. A four bit design is shown since it is easily compared to the Burst designs. The complexity in hardware is directly proportional to the number of bits used.

Weighted binary division require more hardware than the multiplier. The principal addition is a magnitude detector to determine whether subtraction is possible. The divider, of course, uses a subtractor

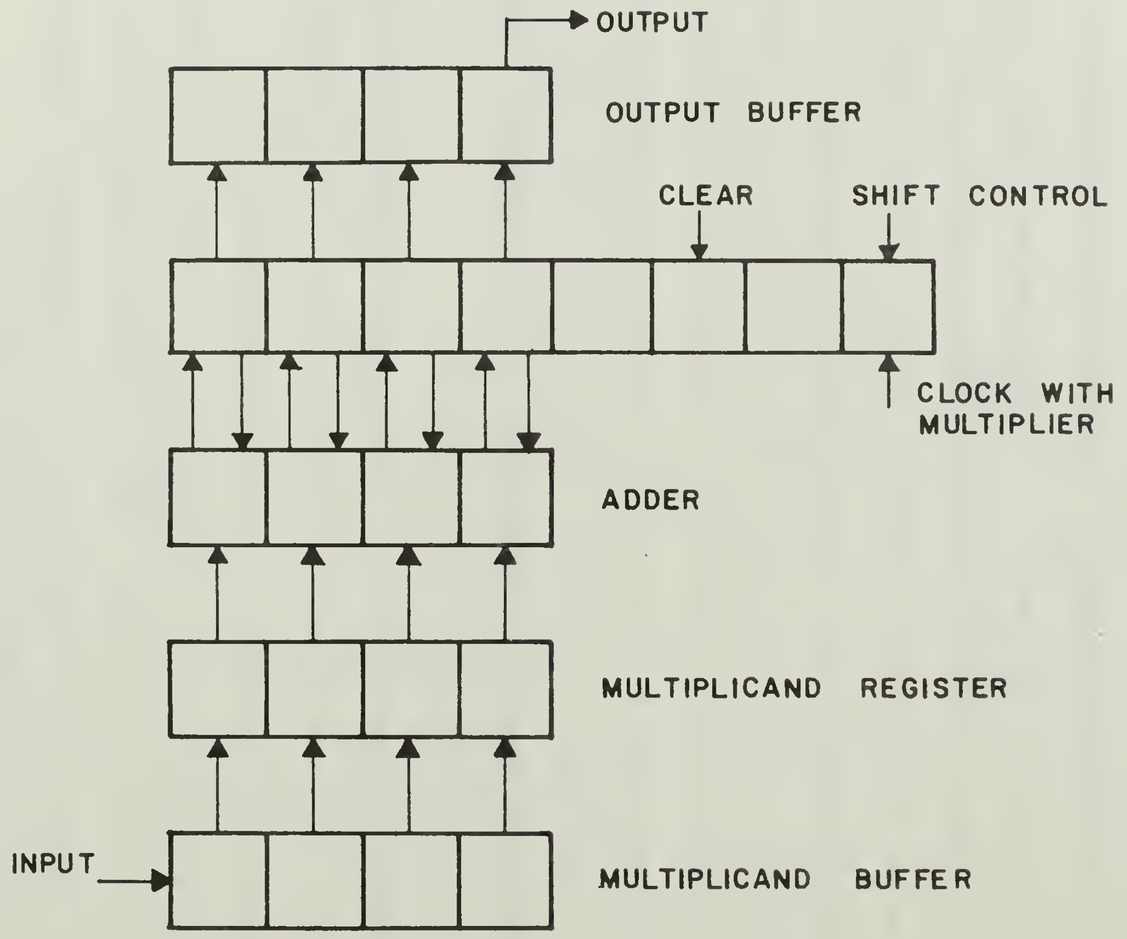


Figure 25. A typical design for a binary multiplier.

in place of the adder.

4.5 Other Arithmetic Processors

The comparison of Bursts to such techniques as the Digital Differential Analyzer (DDA) (6) and the Binary Rate Multiplier (BRM)(7) have been attempted, but because of the specialized uses of these circuits a direct comparison is not possible. The principal function of the DDA and BRM is as an integrator. When these devices are used for simple multiplication or addition, hardware is being under-utilized. To attempt to use Burst hardware to perform integration is difficult because the dynamic range of number representation with Bursts is insufficient.

4.6 Summary

This section consists of a series of tables that compare various features of adders, multipliers and dividers for the various processing techniques discussed.

Table 3: Adder Comparisons

	Weighted Binary	BSR Adder	Sampler Adder	PA
Hardware Complexity (gate equivalents)	6	160	6/20	6
Clock Speed (MHz, TTL)	20	15	15	20
Synchronization Necessary?	yes	no	no	no
Special clock/ control signals	no	no	yes	no
Noise immunity	no	yes	yes	yes
Compacted output	NA	yes	no	no
Notes		1, 4	2, 3	1, 2

1 Inputs may be uncompacted

2 Output Bursts may be compacted with 44 g.e. 10 MHz clock

3 Hardware complexity depends if clock is provided or generated

4 10 bit Bursts

Table 4: Multiplier Comparisons

	4-bit Weighted Binary	BSR Multiplier	Rotating Register Multiplier	PA Multipliers			
				8-bit		10-bit	
				full	low cost	full	low cost
Hardware Complexity (gate equivalents)	105	160	160	70	58	100	78
Clock Speed (MHz, TTL)	6	15	6	6	4	5	3.5
Synchronization Necessary?	yes	no	no	no	no	no	no
Special Clock/ Control signals	yes	no	yes	no	no	no	no
Noise immunity	no	yes	yes	yes	yes	yes	yes
Compacted output	NA	yes	yes	no	no	no	no
Notes		1,2	1	2,3	2,3,4	2,3	2,3,4

1 10-bit Bursts

2 Inputs may be uncompacted

3 Output Bursts may be compacted with 44 g.e. at no speed decrease

4 One flip-flop per level in the PA tree

Table 5: Divider Comparisons

	4-bit Weighted Binary	BSR Divider	Rotating Register Divider	PA Dividers			
				8 bit		10 bit	
				full	low cost	full	low cost
Hardware Complexity (gate equivalents)	115	240	160	86	74	120	98
Clock Speed (MHz, TTL)	6	15	6	5	3.5	4	3
Synchronization Necessary?	yes	no	no	no	no	no	no
Special Clock/ Control signals	yes	no	yes	no	no	no	no
Noise immunity	no	yes	some	yes	yes	yes	yes
Compacted output	NA	yes	yes	no	no	no	no
Notes		1,2	1	2,3	2,3,4	2,3	2,3,4

1 10-bit Bursts

2 Inputs may be uncompacted

3 Output Bursts may be compacted with 44 g.e. at no speed decrease

4 One flip-flop per level in the PA tree

5 CONCLUSIONS

The PA has been shown to be a very competitive (in many respects the best) way to perform arithmetic on data in the Burst format. Even with the large amount of redundancy compared to weighted binary the PA circuits compare favorably in both complexity and speed of operation with weighted binary circuits.

The major advantages over other logic design approaches to Burst processing are the ability to use the data on line and to respond very quickly to changes in that data. An additional advantage is that the PA designs do not ignore any of the incoming data making the assumption that the ignored data is identical to the last data received.

As with all Burst designs other than LOCOBURST, there is the persistent problem that the sum of products does not equal the product of sums. If the rate of change of the input is small, this is not a serious problem.

The PA designs presented are easy to construct because only readily available parts are needed and the parts count is quite low. Since the entire calculation is done with logic the parts matching and adjustment problems of the BSR are eliminated.

PA designs ushered in the concept of the uncompacted Burst, principally because this design was capable of accepting as well as producing data in that format. Uncompacted Bursts have both advantages and disadvantages. The uncompacted format allows the processor to respond to input changes more quickly and is generally cheaper. The principle disadvantage is the loss of the ability to do some error correction before the data is processed (majority function decoding, etc.).

Both sign and magnitude and a biased representation of signal numbers have been investigated. Generalized arithmetic in either system nearly doubles the complexity of the hardware, however, many applications (those requiring only addition/subtraction or only multiplication/division) can be handled with little or no increase in hardware complexity.

The utility of the PA designs has been verified by the students of the Information Engineering Laboratory (IEL) who have chosen or adopted PA designs for projects which required Burst arithmetic processing. These projects provide excellent evidence that this search for a better approach to Burst arithmetic processing has been successful.

REFERENCES

1. Poppelbaum, W. J., Appendix I to "A Practicability Program in Stochastic Processing", proposal for the Office of Naval Research, Department of Computer Science, University of Illinois, March 1974.

2. Annual Progress Report, UIUCDCS-APR-76, Department of Computer Science, University of Illinois, April 1976, pp 1-98.

3. Taylor, G. L., "An Analysis of Burst Encoding Methods and Transmission Properties", M.S. Thesis, UIUCDCS-R-75-770, Department of Computer Science, University of Illinois, December 1975.

4. Kohavi, Z., Switching and Finite Automata Theory, McGraw-Hill, New York, 1970, pp 241-263.

5. Bracha, E., "BURSTCALC (A BURST CALCulator)", M.S. Thesis, UIUCDCS-R-75-769, Department of Computer Science, University of Illinois, October 1975.

6. Sizer, T.R.H., The Digital Differential Analyser, Chapman and Hall, London, 1968.

7. Newman, W.M. and R.F. Sproull, Principles of Interactive Computer Graphics, McGraw-Hill, New York, 1973, pp 41-48.

Appendix A: ICs Used in the CPU

Board 1

Position #	Type	Description
1	7432	Quad 2-input OR
2	7432	Quad 2-input OR
3	7430	8-input NAND
4	7430	8-input NAND
5	----	
6	74121	Monostable Flip-flop
7	74163	4-bit Counter
8	7474	Dual D Flip-flop
9	74164	8-bit Parallel-out SR
10	7430	8-input NAND
11	7432	Quad 2-input OR
12	7432	Quad 2-input OR
13	7446	7-segment Decoder/Driver
14	7475	4-bit Latch
15	7490	Decade Counter
16	7402	Quad 2-input NOR
17	7404	Hex inverter
18	74163	4-bit Counter
19	7473	Dual J-K Flip-flop
20	7451	Dual 2-wide 2-input AOI
21	7405	Open-collector Hex Inverter
22	7490	Decade Counter
23	7475	4-bit Latch
24	7446	7-segment Decoder/Driver

Board 2

Position #	Type	Description
1	7486	Quad 2-input XOR
2	7430	8-input NAND
3	74164	8-bit Parallel-out SR
4	7430	8-input NAND
5	7430	8-input NAND
6	7430	8-input NAND
7	7400	Quad 2-input NAND
8	7400	Quad 2-input NAND
9	7420	Dual 4-input NAND
10	7430	8-input NAND
11	7400	Quad 2-input NAND
12	7400	Quad 2-input NAND
13	7400	Quad 2-input NAND
14	7474	Dual D Flip-flop
15	74H183	Dual Full Adder
16	7474	Dual D Flip-flop
17	74H183	Dual Full Adder
18	----	
19	----	
20	74H183	Dual Full Adder
21	7474	Dual D Flip-flop
22	74H183	Dual Full Adder
23	7474	Dual D Flip-flop
24	7402	Quad 2-input NOR

Appendix B: Edge Connector Pin Assignments

Board 1

Pin #	Description
1-8	Switches for "A" Burst
9-16	Switches for "B" Burst
17	Master clock from selector
18	Display format switch
19	DATA from CPU board
20	Clock to CPU
30	A.C. ripple from power supply
31	Clock 1 to selector
32	Clock 2 to selector
33	Clock 3 to selector
34	Clock 4 to selector
35	Clock 5 to selector
60	B Burst to CPU
61	A Burst to CPU
64	Wait Light
65-71	Seven segment units display
72-78	Seven segment tens display
79	Ground
80	V_{cc} (+5 volts)

Board 2

Pin #	Description
1	CPU clock in
2	Mode select switch
3	Burst A in
4	Burst B in
78	$\overline{\text{DATA}}$ result out
79	Ground
80	V_{cc} (+5 volts)

VITA

Leon Clemens Tietz was born February 28, 1948 in New Prague, Minnesota. He graduated from New Prague High School in 1966, and entered Mankato State College, Mankato, Minnesota. In 1970 he graduated Cum Laude with a B.A. in Physics, and in 1972 he entered the University of Illinois to pursue a Ph.D. in Computer Science.

While attending the University of Ilinois he was employed by the U.S. Army Construction Engineering Research Laboratory (CERL), Champaign, Illinois as a graduate research assistant. He is currently a permanent employee with the same organization. He has co-authored several published reports while employed by CERL.

He is a member of the Association for Computing Machinery and the Institute of Electrical and Electronics Engineers. He is a member of Sigma Zeta and an associate member of Sigma Xi honorary fraternities.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER UIUCDCS-R-77-895	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) BURSTLOGIC: DESIGN AND ANALYSIS OF LOGIC TO PERFORM ARITHMETIC ON DATA IN THE BURST FORMAT		5. TYPE OF REPORT & PERIOD COVERED Ph.D. Thesis, May, 1977
		6. PERFORMING ORG. REPORT NUMBER UIUCDCS-R-77-8895
7. AUTHOR(s) Leon Clemens Tietz		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0982
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Code 437 Arlington, Virginia 22217		12. REPORT DATE May 1977
		13. NUMBER OF PAGES 65
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Release Unlimited
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Perverted Adder
Burst Processing
Burst Arithmetic

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A family of Processors has been developed to perform arithmetic on data in the Burst Format. The common building block of these circuits is a simple finite state machine called the Perverted Adder (PA) which performs BURST addition. The interconnection of PAs in a tree formation with some additional circuitry produces a Burst multiplier or divider. A PA CPU has been constructed to demonstrate these PA designs.

LIOGRAPHIC DATA EET	1. Report No. UIUCDCS-R-77-895	2.	3. Recipient's Accession No.

itle and Subtitle
URSTLOGIC: DESIGN AND ANALYSIS OF LOGIC TO PERFORM
RITHMETIC ON DATA IN THE BURST FORMAT

5. Report Date
May 1977

6.

uthor(s)
eon Clemens Tietz

8. Performing Organization Rept. No. UIUCDCS-77-895

erforming Organization Name and Address
epartment of Computer Science
niversity of Illinois at Urbana-Champaign
rbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.
N00014-75-C-0982

Sponsoring Organization Name and Address
ffice of Naval Research
ode 437
rlington, Virginia 22217

13. Type of Report & Period Covered
Ph.D.

14.

Supplementary Notes

Abstracts

A family of Processors has been developed to perform arithmetic on data in the urst Format. The common building block of these circuits is a simple finite state achine called the Perverted Adder (PA) which performs BURST addition. The inter- onnection of PAs in a tree formation with some additional circuitry produces a urst multiplier or divider. A PA CPU has been constructed to demonstrate these A designs.

Key Words and Document Analysis. 17a. Descriptors

erverted Adder
urst Processing
urst Arithmetic

Identifiers/Open-Ended Terms

COSATI Field/Group

vailability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 65
lease Unlimited	20. Security Class (This Page UNCLASSIFIED	22. Price

BLIOGRAPHIC DATA EET	1. Report No. UIUCDCS-R-77-895	2.	3. Recipient's Accession No.

Title and Subtitle
BURSTLOGIC: DESIGN AND ANALYSIS OF LOGIC TO PERFORM
ARITHMETIC ON DATA IN THE BURST FORMAT

5. Report Date
May 1977

6.

Author(s)
Leon Clemens Tietz

8. Performing Organization Rept. No. UIUCDCS-77-895

Performing Organization Name and Address
Department of Computer Science
Jniversity of Illinois at Urbana-Champaign
Jrbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.
N00014-75-C-0982

Sponsoring Organization Name and Address
)ffice of Naval Research
Code 437
Arlington, Virginia 22217

13. Type of Report & Period Covered
Ph.D.

14.

Supplementary Notes

Abstracts

A family of Processors has been developed to perform arithmetic on data in the Burst Format. The common building block of these circuits is a simple finite state machine called the Perverted Adder (PA) which performs BURST addition. The interconnection of PAs in a tree formation with some additional circuitry produces a burst multiplier or divider. A PA CPU has been constructed to demonstrate these PA designs.

Key Words and Document Analysis. 17a. Descriptors

erverted Adder
urst Processing
urst Arithmetic

. Identifiers/Open-Ended Terms

COSATI Field/Group

Availability Statement Release Unlimited	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 65
	20. Security Class (This Page UNCLASSIFIED	22. Price

LIOGRAPHIC DATA ET	1. Report No. UIUCDCS-R-77-895	2.	3. Recipient's Accession No.
itle and Subtitle JRSTLOGIC: DESIGN AND ANALYSIS OF LOGIC TO PERFORM RITHMETIC ON DATA IN THE BURST FORMAT			5. Report Date May 1977
			6.
uthor(s) eon Clemens Tietz			8. Performing Organization Rept. No. UIUCDCS-77-895
erforming Organization Name and Address epartment of Computer Science niversity of Illinois at Urbana-Champaign rbana, Illinois 61801			10. Project/Task/Work Unit No.
			11. Contract/Grant No. N00014-75-C-0982
ponsoring Organization Name and Address ffice of Naval Research de 437 rlington, Virginia 22217			13. Type of Report & Period Covered Ph.D.
			14.
upplementary Notes			

bstracts
A family of Processors has been developed to perform arithmetic on data in the rst Format. The common building block of these circuits is a simple finite state chine called the Perverted Adder (PA) which performs BURST addition. The inter-nnection of PAs in a tree formation with some additional circuitry produces a rst multiplier or divider. A PA CPU has been constructed to demonstrate these designs.

Key Words and Document Analysis. 17a. Descriptors

rverted Adder
rst Processing
rst Arithmetic

Identifiers/Open-Ended Terms

COSATI Field/Group

vailability Statement lease Unlimited	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 65
	20. Security Class (This Page UNCLASSIFIED	22. Price